

UC Berkeley - CS267

Building a Reliable Software Infrastructure for Scientific Computing

Osni Marques

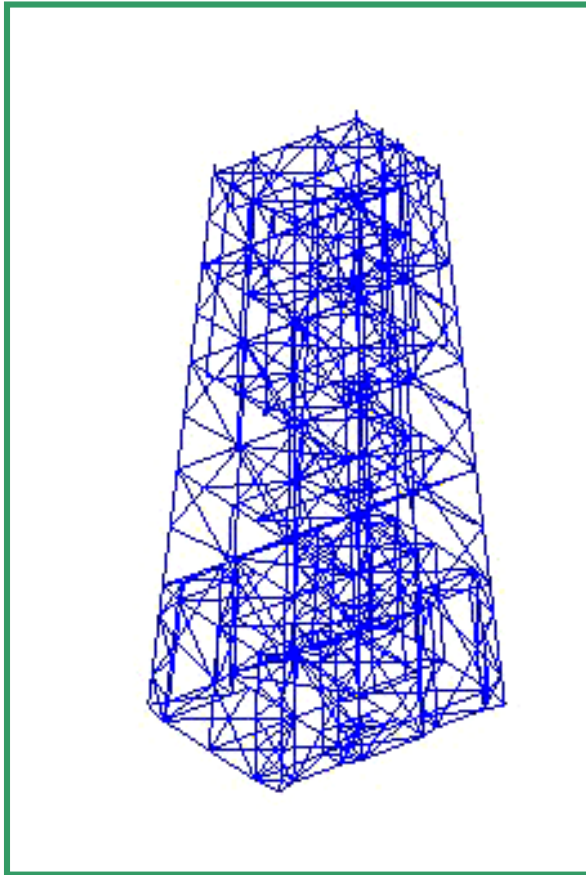
Lawrence Berkeley National Laboratory (LBNL)

oamarques@lbl.gov

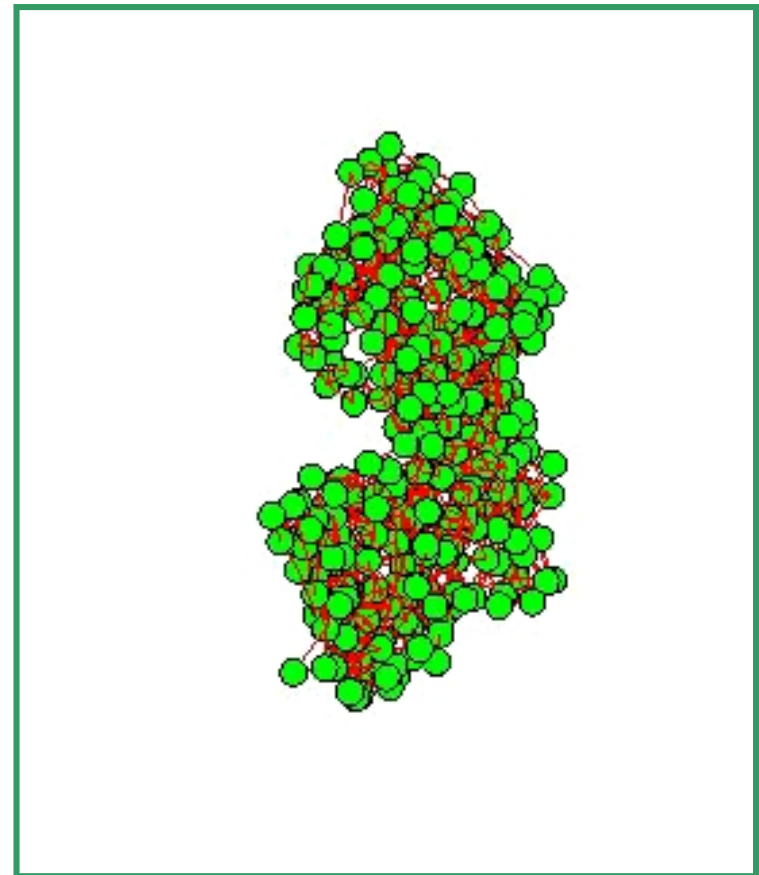
Outline

- Keeping the pace with the software and hardware
 - Hardware evolution
 - Performance tuning
 - Software selection
 - What is missing?
- The DOE ACTS Collection Project
 - Goals
 - Related activities
 - Current features
 - Lessons learned

Two Applications: $Ax=\lambda x$

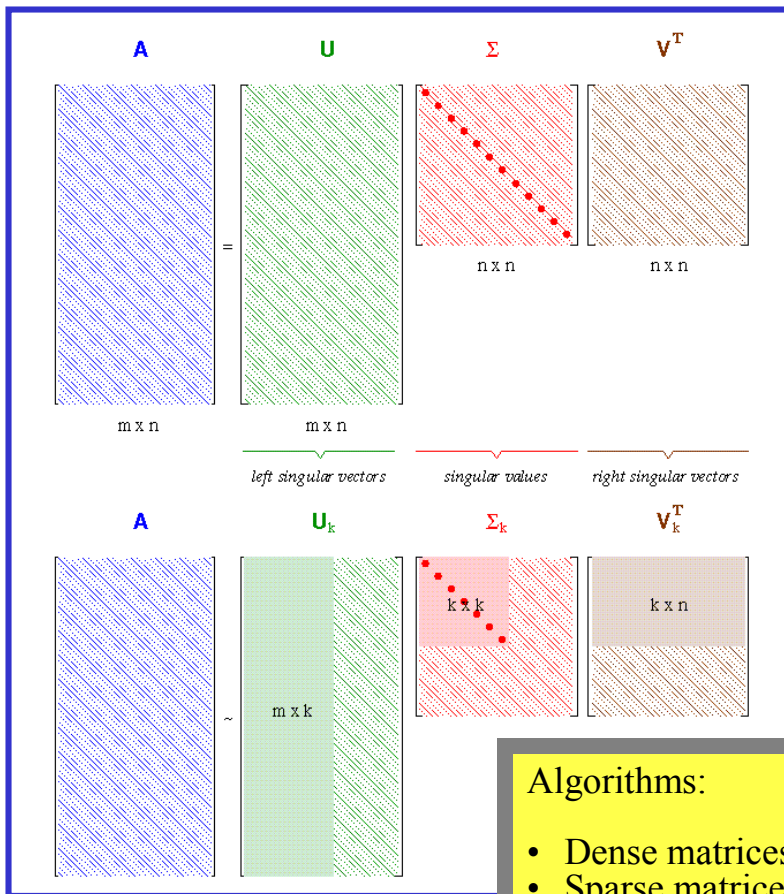


Simplified model of the “Namorado” steel jacket platform, 170 meters water depth, 2463 degrees of freedom, $\omega_1=1.71\text{rad/s}$.



Simplified model of the maltodextrin protein (370 residues) using the RTB method.

Singular Value Decomposition (SVD)



Applications:

- Digital signal processing
- Protein substate modeling and identification
- Spectrum analysis
- Model reduction
- Fuzzy and neural systems
- Data compression
- Inverse problems
- Information retrieval
- \vdots

Algorithms:

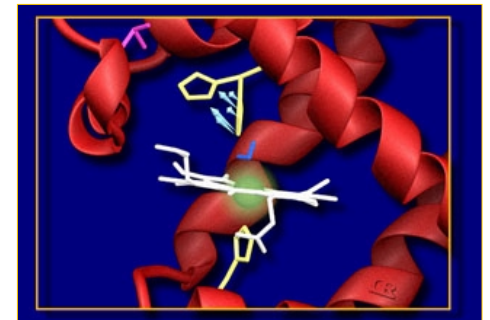
- Dense matrices: LAPACK, ScaLAPACK,...
- Sparse matrices: Subspaces, Lanczos,...

SVD and eigenvalues:

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} u \\ v \end{bmatrix}, \quad \lambda = \pm \sigma$$

$$(AA^T)u = \lambda u, \quad \lambda = \sigma^2$$

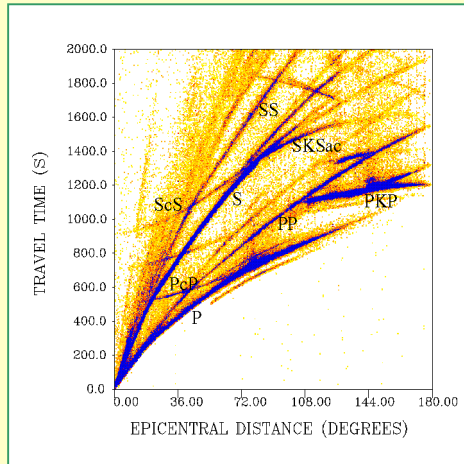
$$(A^T A)v = \lambda v, \quad \lambda = \sigma^2$$



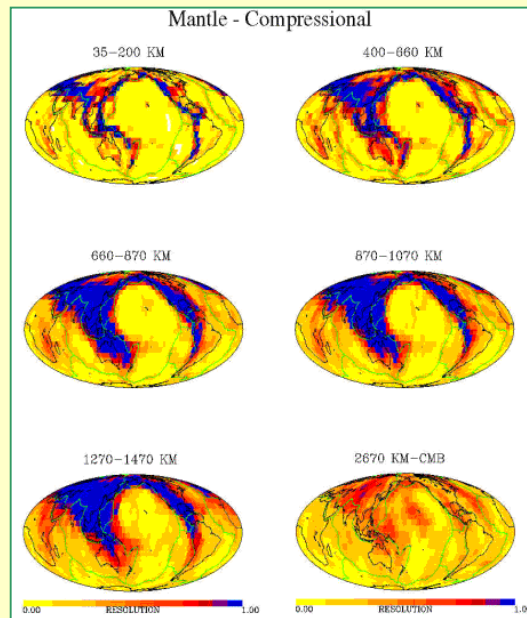
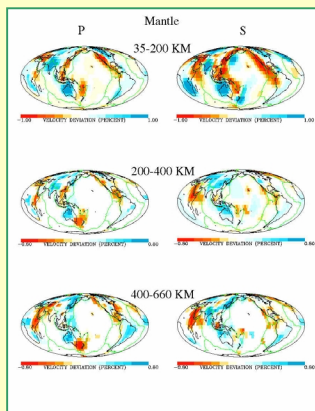
<http://www-bioc.rice.edu/~tromo>

Two SVD Applications

$$\delta t_{ijk} = \sum_{l=1}^{n_{3D}} \frac{\partial t_{ijk}}{\partial v^l} \delta v^l_k + \sum_{l=1}^4 \frac{\partial t_{ijk}}{\partial v^l} \delta h^l_j + \sum_{l=1}^{n_s} \frac{\partial t_{ijk}}{\partial s^l} \delta s^l_{ik} + \sum_{l=1}^{n_b} \frac{\partial t_{ijk}}{\partial b^l} \delta b^l + \sum_{l=1}^{n_a} \frac{\partial t_{ijk}}{\partial a^l} \delta a^l$$



- **Data:** travel times of sound waves generated by earthquakes used to infer structure in the entire Earth (crust, mantle and core).
- **Goal:** model for the internal structure of the Earth.
- More than 1×10^6 data points and 2×10^5 parameters.



United States Patent [19]
Landauer et al.

US005301109A
[11] Patent Number: 5,301,109
[45] Date of Patent: Apr. 5, 1994

[54] COMPUTERIZED CROSS-LANGUAGE DOCUMENT RETRIEVAL USING LATENT SEMANTIC INDEXING

[75] Inventors: Thomas K. Landauer, Summit; Michael L. Littman, Somerset, both of N.J.

[73] Assignee: Bell Communications Research, Inc., Livingston, N.J.

[21] Appl. No.: 734,291

[22] Filed: Jul. 17, 1991

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 536,029, Jun. 11, 1990, abandoned.

[51] Int. Cl.² G06F 15/40

[52] U.S. Cl. 364/419.19; 364/419.16

[58] Field of Search 364/419

References Cited

U.S. PATENT DOCUMENTS

4,270,182 5/1981 Asija 364/900

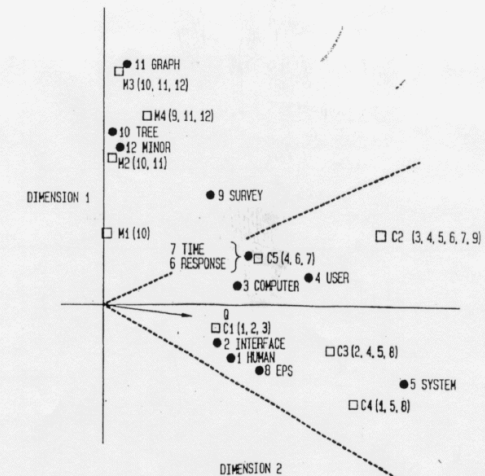
4,593,356 6/1986 Hashimoto et al. 364/419
4,654,798 3/1987 Taki et al. 364/419
4,839,853 6/1980 Deewester et al. 364/900
4,864,503 9/1989 Tolin 364/419
4,916,614 4/1990 Koji et al. 364/900
5,128,865 7/1992 Sadler 364/419

Primary Examiner—Roy N. Envall, Jr.
Assistant Examiner—Xuong Chang
Attorney, Agent, or Firm—Leonard Charles Suchyia;
Joseph Giordano

[57] ABSTRACT

A methodology for retrieving textual data objects in a multiplicity of languages is disclosed. The data objects are treated in the statistical domain by presuming that there is an underlying, latent semantic structure in the usage of words in each language under consideration. Estimates to this latent structure are utilized to represent and retrieve objects. A user query is recoded in the new statistical domain and then processed in the computer system to extract the underlying meaning to respond to the query.

8 Claims, 3 Drawing Sheets

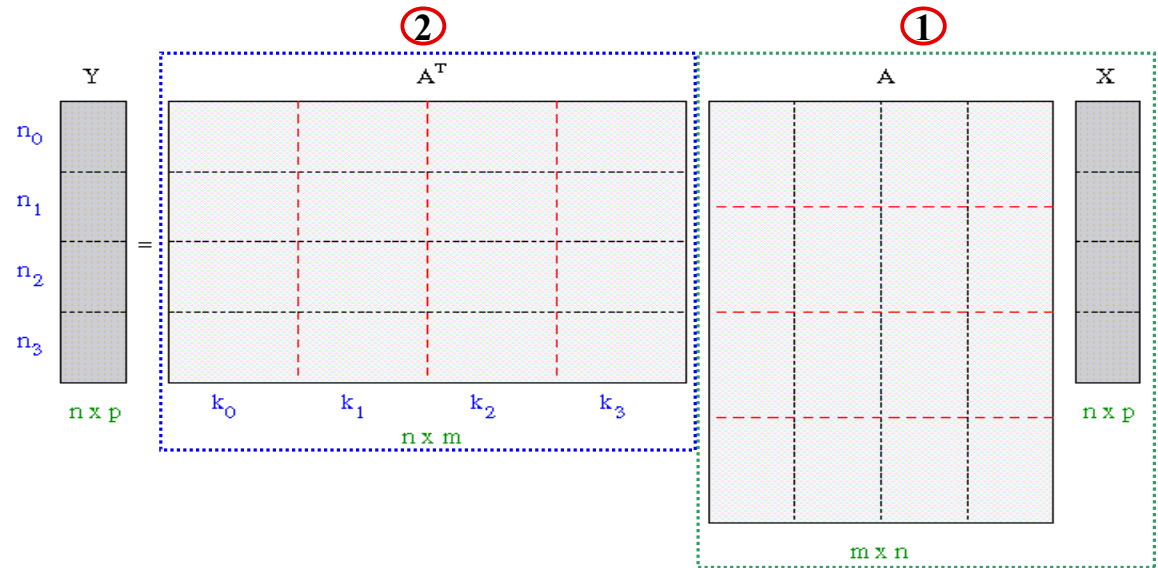


<http://www.cs.utk.edu/~lsi/>

Computing $Y=(A^T A)X$ in Parallel

Lanczos: $(A^T A)v = \lambda v$

- Requires only products of the form $r = Hq$
- Generates a basis $Q = [q_1 \ q_2 \ \dots \ q_j]$ through a simple three-term recurrence
- The projection of H into Q is a symmetric tridiagonal matrix T (of dimension $j \ll n$)
- The solutions of $Ts = \theta s$ together with Q lead to approximate solutions of (λ, x)



- One processor reads A (by rows) and sends the data to other processors
- A is stored by means of 3 arrays:
 - one (integer) array stores the number of nonzero entries in each row
 - one (integer) array stores the column indices
 - one (real) array stores the corresponding entries
- A redistribution of A is performed so as each processor contains roughly the same number of entries
- A quick sort is applied to the indices to prevent cache misses

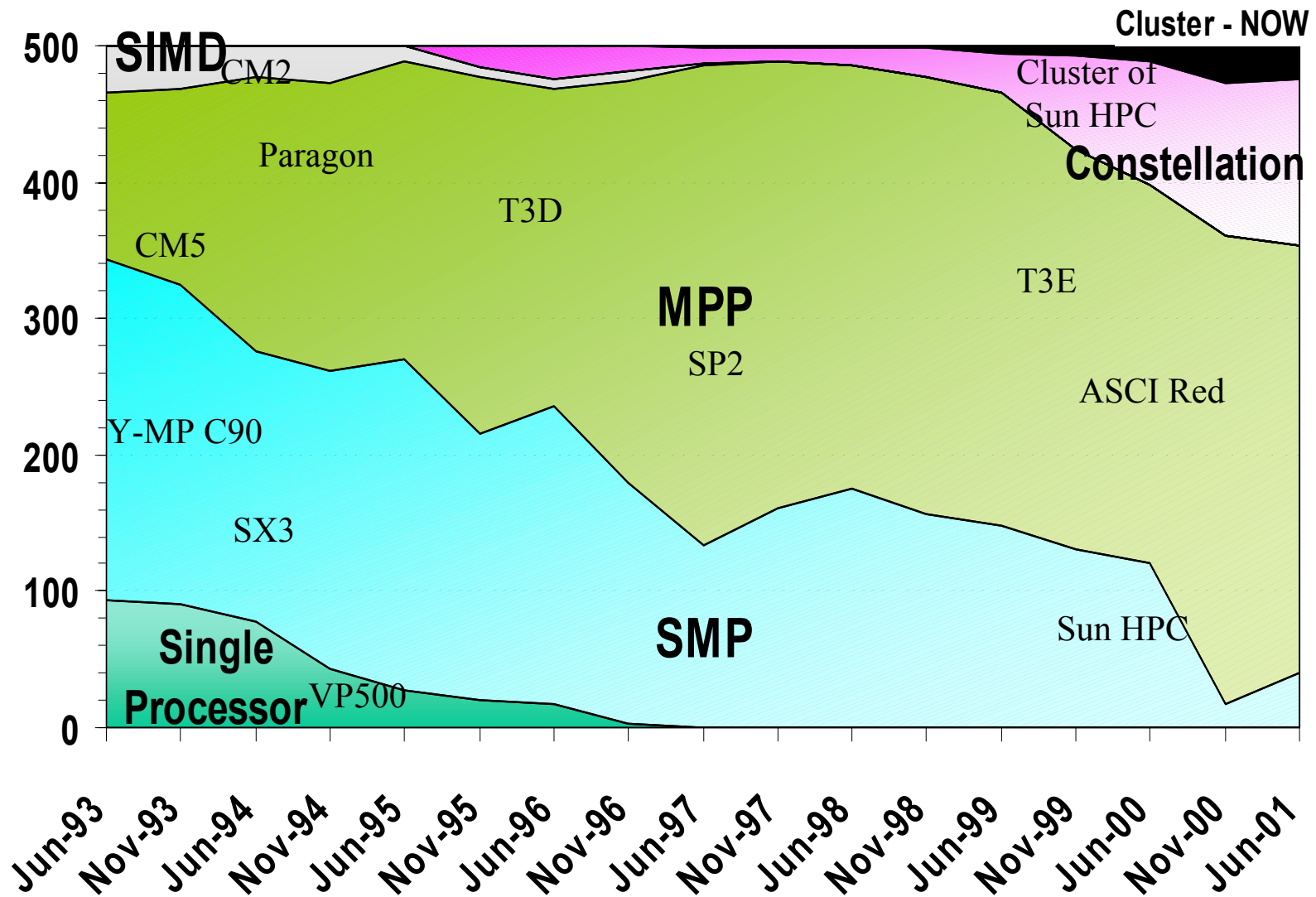
Model of dimension 846968 by 96300, 28587210 nonzeros,
 $k=500$, $nvb=1$ to 5, $tol=10^{-6}$, 32 processors (Cray T3E 900).

time	block size				
	1	2	3	4	5
total (s)	1270	1190	1110	1100	1170
$(A^T A)x$	49.6% (1293)	50.5% (671)	53.5% (460)	54.3% (352)	52.6% (289)
basis generation	0.1%	0.2%	0.3%	0.4%	0.8%
reorthogonalization	12.2%	11.5%	8.1%	7.7%	7.1%
reduced problem (T)	19.7%	16.4%	15.0%	13.1%	13.2%
eigenvectors	18.3%	21.4%	20.0%	24.5%	26.3%

High Performance Computers (Sustainable Performance)

- ~ 20 years ago → 1×10^6 Floating Point Ops/sec (Mflop/s)
 - Scalar based
- ~ 10 years ago → 1×10^9 Floating Point Ops/sec (Gflop/s)
 - Vector & Shared memory computing, bandwidth aware
 - Block partitioned, latency tolerant
- ~ Today → 1×10^{12} Floating Point Ops/sec (Tflop/s)
 - Highly parallel, distributed processing, message passing, network based
 - data decomposition, communication/computation
- ~ 10 years away → 1×10^{15} Floating Point Ops/sec (Pflop/s)
 - Many more levels of memory hierarchy, combination of grids&HPC
 - More adaptive, latency and bandwidth aware, fault tolerant, extended precision, attention to SMP nodes

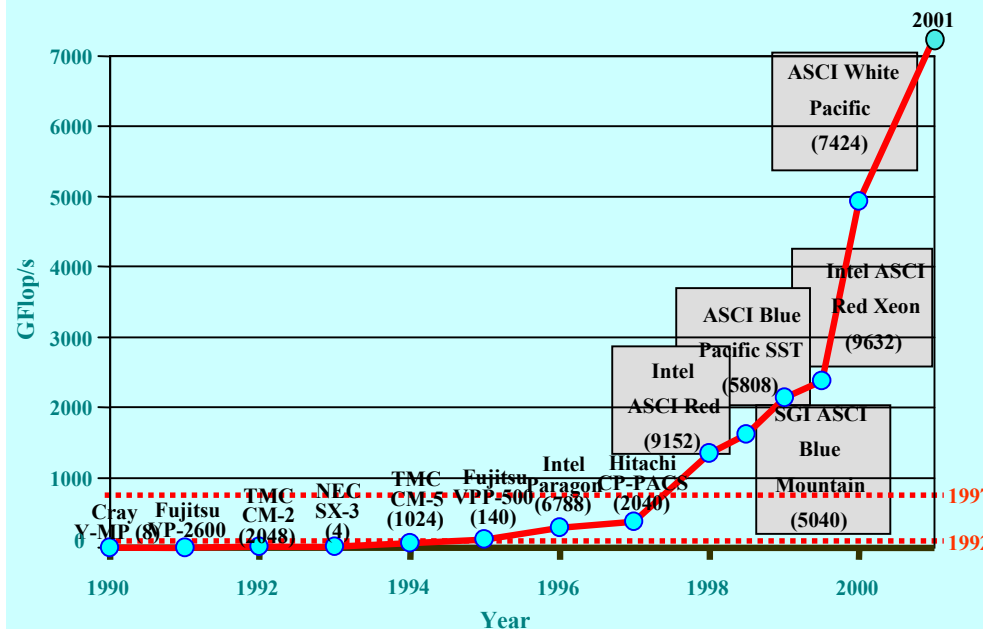
Architectures



Why do we need these tools?

Top 5 Machines:
<http://www.top500.org>
 (June 2003)

Rank	Manufacturer	Computer	R_{max} [TF/s]	Installation Site	Country	Year	Area of Installation	# Proc
1	NEC	Earth-Simulator	35.86	Earth Simulator Center	Japan	2002	Research	5120
2	HP	ASCI Q, AlphaServer SC	13.88	Los Alamos National Laboratory	USA	2002	Research	8192
3	Linux NetworX	MCR Linux Cluster Xeon	7.63	Lawrence Livermore National Laboratory	USA	2002	Research	2304
4	IBM	ASCI White SP Power3	7.30	Lawrence Livermore National Laboratory	USA	2000	Research	8192
5	IBM	SP Power3	7.30	NERSC/LBNL	USA	2002	Research	6656



A computation that took 1 full year to complete in 1980 could be done in ~ 10 hours in 1992, in ~ 16 minutes in 1997 and in ~ 27 seconds in 2001!

- High Performance Tools
 - portable
 - library calls
 - robust algorithms
 - help code optimization
- More code development in less time!
- More simulation in less computer time!

Automatic Tuning

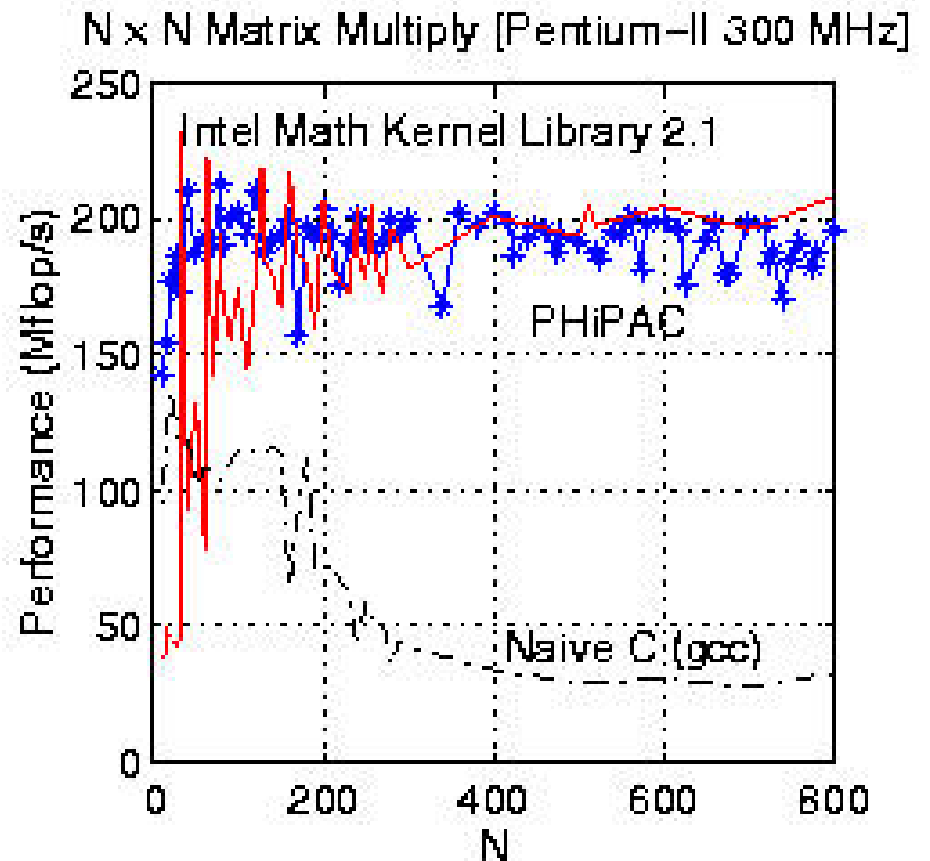
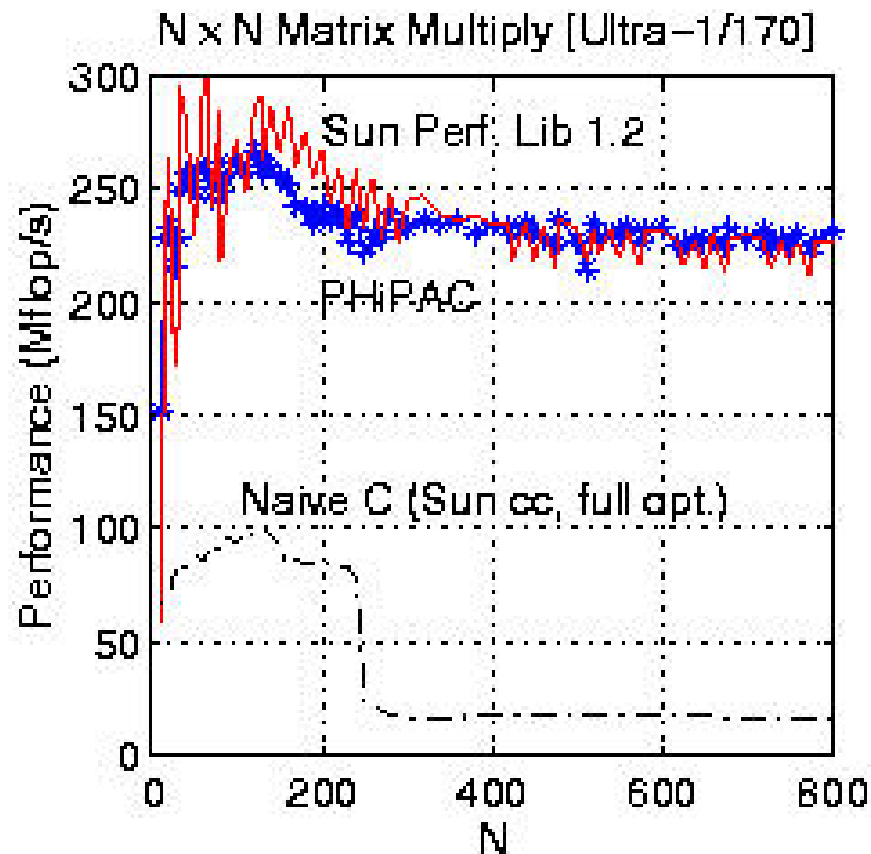
- For each kernel
 1. Identify and generate a space of algorithms
 2. Search for the fastest one, by running them
- What is a space of algorithms?
 - Depending on kernel and input, may vary
 - instruction mix and order
 - memory access patterns
 - data structures
 - mathematical formulation
- When do we search?
 - Once per kernel and architecture
 - At compile time
 - At run time
 - All of the above

- PHiPAC:
www.icsi.berkeley.edu/~bilmes/hipac
- ATLAS:
www.netlib.org/atlas
- XBLAS:
www.nersc.gov/~xiaoye/XBLAS
- Sparsity: www.cs.berkeley.edu/~yelick/sparsity
- FFTs and Signal Processing
 - FFTW: www.fftw.org
 - Won 1999 Wilkinson Prize for Numerical Software
 - SPIRAL: www.ece.cmu.edu/~spiral
 - Extensions to other transforms, DSPs
 - UHFFT
 - Extensions to higher dimension, parallelism

Tuning pays off!

Example: *PHiPAC* \Rightarrow

$$C = A * B$$

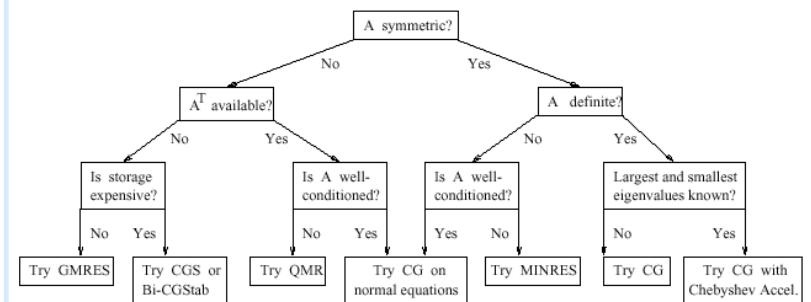


What About Software Selection?

Example: $Ax = b$

- Use a direct solver ($A=LU$) if
 - Time and storage space acceptable
 - Iterative methods don't converge
 - Many b 's for same A
- Criteria for choosing a direct solver
 - Symmetric positive definite (SPD)
 - Symmetric
 - Symmetric-pattern
 - Unsymmetric
- Row/column ordering schemes available
 - MMD, AMD, ND, graph partitioning
- Hardware

Build a preconditioning matrix K such that $Kx=b$ is much easier to solve than $Ax=b$ and K is somehow “close” to A (incomplete LU decompositions, sparse approximate inverses, polynomial preconditioners, preconditioning by blocks or domains, element-by-element, etc). See *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*.



Bugs...



On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dharan, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck an American Army barracks, killing 28 soldiers and injuring around 100 other people. The problem was an inaccurate calculation of the time since boot due to computer arithmetic errors.



On June 4, 1996, an Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing \$7 billion. The problem was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer.



On August 23, 1991, the first concrete base structure for the Sleipner A platform sprang a leak and sank under a controlled ballasting operation during preparation for deck mating in Gandsfjorden outside Stavanger, Norway. The post accident investigation traced the error to inaccurate finite element approximation of the linear elastic model of the tricell (using the popular finite element program NASTRAN). The shear stresses were underestimated by 47% leading to insufficient design. In particular, certain concrete walls were not thick enough.

<http://www.zenger.informatik.tu-muenchen.de/persons/huckle/bugse.html>

Underflows and Performance of the dqds Algorithm

function xLASQ1

function xLASQ2

function xLASQ3

function xLASQ4

function xLASQ5

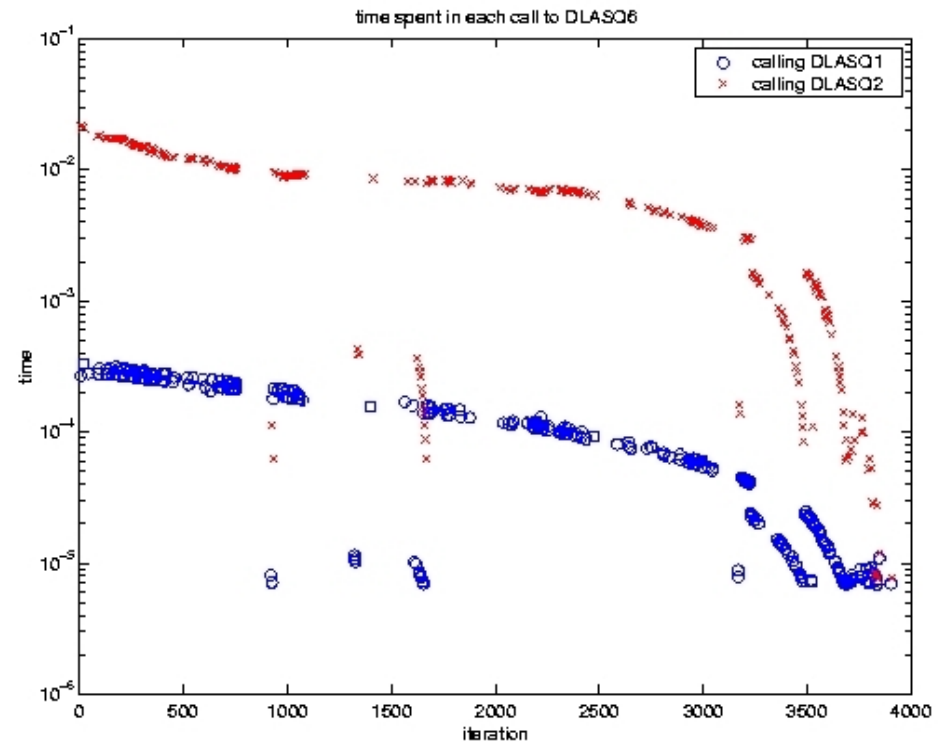
function xLASQ6

Data preprocessing

```

:
IF( Z( J4-2 ).EQ.ZERO ) THEN
  Z( J4 ) = ZERO
  D = Z( J4+1 )
  DMIN = D
  EMIN = ZERO
ELSE IF( SAFMIN*Z( J4+1 ).LT.Z( J4-2 ) .AND.
$      SAFMIN*Z( J4-2 ).LT.Z( J4+1 ) ) THEN
  TEMP = Z( J4+1 ) / Z( J4-2 )
  Z( J4 ) = Z( J4-1 )*TEMP
  D = D*TEMP
ELSE
  Z( J4 ) = Z( J4+1 )*( Z( J4-1 ) / Z( J4-2 ) )
  D = Z( J4+1 )*( D / Z( J4-2 ) )
END IF
:

```



Performance of the dqds algorithm on a SUN Ultra 30 calling xLASQ1 (blue) or xLASQ2 (red), multiplying the same input data by two different scaling factors. Depending on the scaling, a test with SAFMIN (see code to the left) led to a large number of underflows (dealt with at software level and greatly degrading the computational performance).

LAPACK: <http://www.netlib.org/lapack>

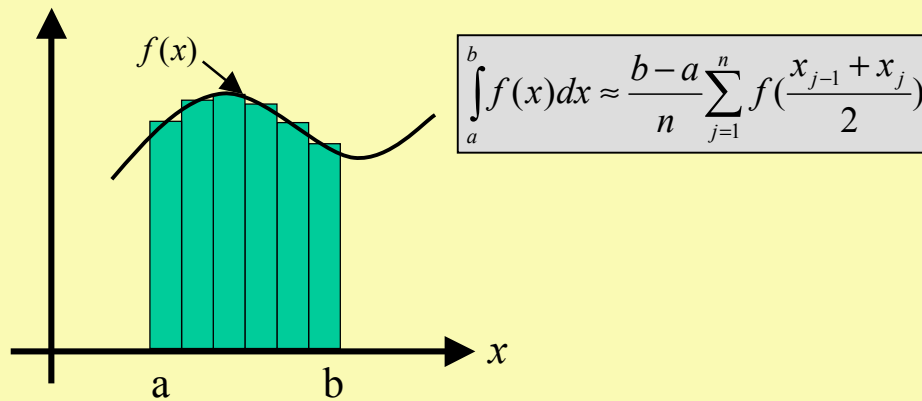


Challenges in the Development of Scientific Codes

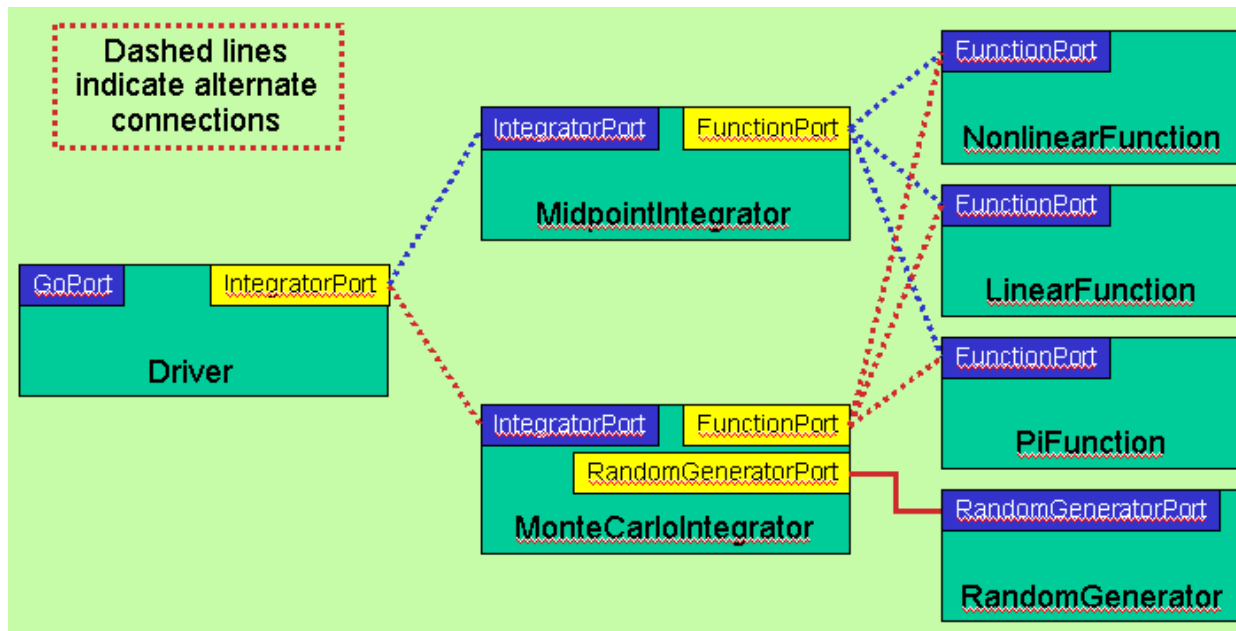
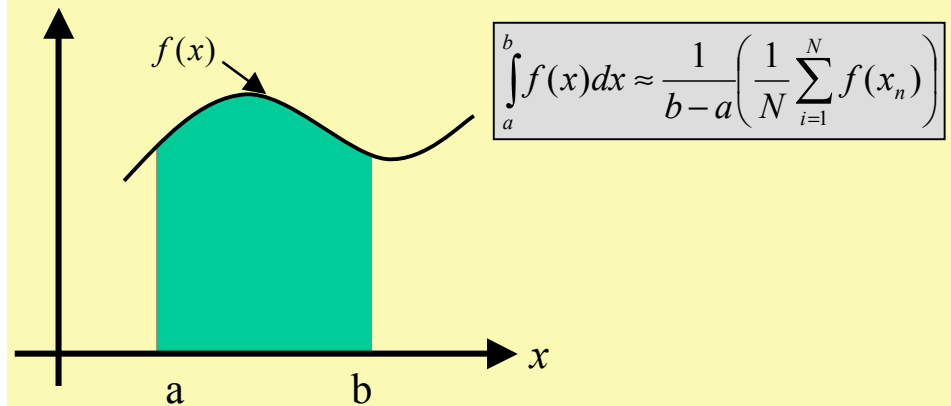
- Productivity
 - Time to the first solution (prototype)
 - Time to solution (production)
 - Other requirements
 - Complexity
 - Increasingly sophisticated models
 - Model coupling
 - Interdisciplinarity
 - Performance
 - Increasingly complex algorithms
 - Increasingly complex architectures
 - Increasingly demanding applications
- Libraries written in different languages.
 - Discussions about standardizing interfaces are often sidetracked into implementation issues.
 - Difficulties managing multiple libraries developed by third-parties.
 - Need to use more than one language in one application.
 - The code is long-lived and different pieces evolve at different rates
 - Swapping competing implementations of the same idea and testing without modifying the code
 - Need to compose an application with some other(s) that were not originally designed to be combined

Components: *simple example*

Numerical integration: midpoint



Numerical integration: Monte Carlo



$$f_1(x) = x^2$$

$$f_2(x) = 2x$$

$$f_3(x) = \frac{4}{1+x^2}$$

The Reality...

- The development of complex simulation codes on parallel computers is not a trivial task.
- Usually, a significant percentage of the efforts focus on the development of the codes and their optimization.
- There is a need for a collaboration framework for ongoing development and deployment of computational tools.
- In 1999, the PITAC Report recommended the creation of a national library of certified domain-specific software in order to reduce the labor required for software development, testing and evolution.
- Research in computational sciences is fundamentally interdisciplinary and addresses, among many others, climate and environment modeling, DNA sequencing, flows in geological structures, etc.

What is the ACTS Collection?

<http://acts.nersc.gov>

- Advanced CompuTational Software Collection
- Tools for developing parallel applications
- ACTS started as an “umbrella” project

Goals

- ❑ *Extended support for experimental software*
- ❑ *Make ACTS tools available on DOE computers*
- ❑ *Provide technical support (acts-support@nersc.gov)*
- ❑ *Maintain ACTS information center (<http://acts.nersc.gov>)*
- ❑ *Coordinate efforts with other supercomputing centers*
- ❑ *Enable large scale scientific applications*
- ❑ *Educate and train*

ACTS: *levels of support*

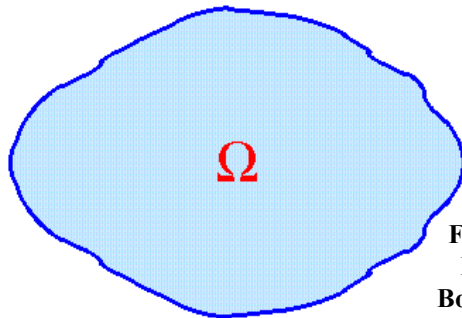
- ***High***
 - Intermediate level
 - Tool expertise
 - Conduct tutorials
- ***Intermediate***
 - Basic level
 - Provide a higher level of support to users of the tool
- ***Basic***
 - Basic knowledge of the tools
 - Help with installation
 - Compilation of user's reports (*acts-support@nersc.gov*)

ACTS Tools Functionalities

Category	Tool	Functionalities
Numerical $Ax = b$ $Az = \lambda z$ $A = U\Sigma V^T$ PDEs ODEs ⋮	Aztec	Algorithms for the iterative solution of large sparse linear systems.
	Hypre	Algorithms for the iterative solution of large sparse linear systems, intuitive grid-centric interfaces, and dynamic configuration of parameters.
	PETSc	Tools for the solution of PDEs that require solving large-scale, sparse linear and nonlinear systems of equations.
	OPT++	Object-oriented nonlinear optimization package.
	SUNDIALS	Solvers for the solution of systems of ordinary differential equations, nonlinear algebraic equations, and differential-algebraic equations.
	ScaLAPACK	Library of high performance dense linear algebra routines for distributed-memory message-passing.
	SuperLU	General-purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations.
	TAO	Large-scale optimization software, including nonlinear least squares, unconstrained minimization, bound constrained optimization, and general nonlinear optimization.
Code Development	Global Arrays	Library for writing parallel programs that use large arrays distributed across processing nodes and that offers a shared-memory view of distributed arrays.
	Overture	Object-Oriented tools for solving computational fluid dynamics and combustion problems in complex geometries.
Code Execution	CUMULVS	Framework that enables programmers to incorporate fault-tolerance, interactive visualization and computational steering into existing parallel programs
	Globus	Services for the creation of computational Grids and tools with which applications can be developed to access the Grid.
	PAWS	Framework for coupling parallel applications within a component-like model.
	SILOON	Tools and run-time support for building easy-to-use external interfaces to existing numerical codes.
	TAU	Set of tools for analyzing the performance of C, C++, Fortran and Java programs.
Library Development	ATLAS and PHiPAC	Tools for the automatic generation of optimized numerical software for modern computer architectures and compilers.
	PETE	Extensible implementation of the expression template technique (C++ technique for passing expressions as function arguments).

Use of ACTS Tools

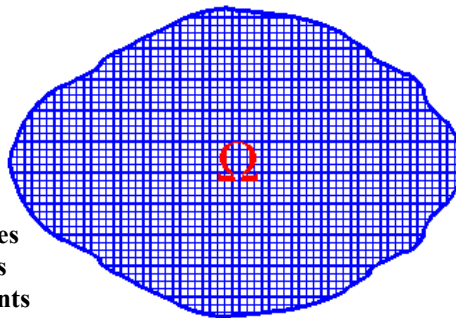
continuous problem



grid

Finite Differences
Finite Elements
Boundary Elements
Fourier
⋮

discrete problem



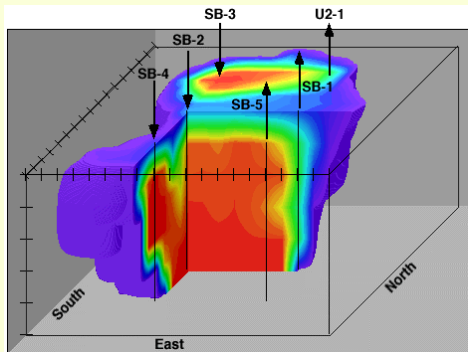
$$m \frac{d^2 x}{dt^2} + c \frac{dx}{dt} + kx = f(t)$$

$$\frac{\partial v}{\partial t} = -v \frac{\partial v}{\partial x} + \Gamma \frac{\partial^2 v}{\partial x^2}$$

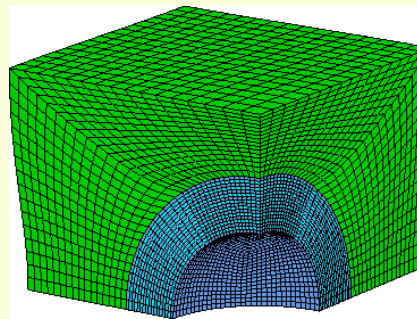
$$\vdots$$

$$= A$$

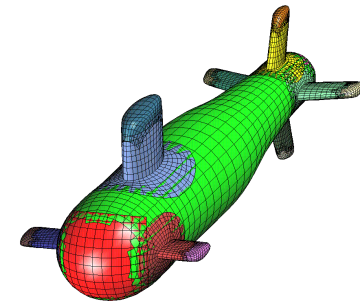
(sparse matrix)



Multiphase flow using *PETSc*, 4 million cell blocks, 32 million DOF, over 10.6 Gflops on an IBM SP (128 nodes), entire simulation runs in less than 30 minutes (Pope, Gropp, Morgan, Seperhrnoori, Smith and Wheeler).

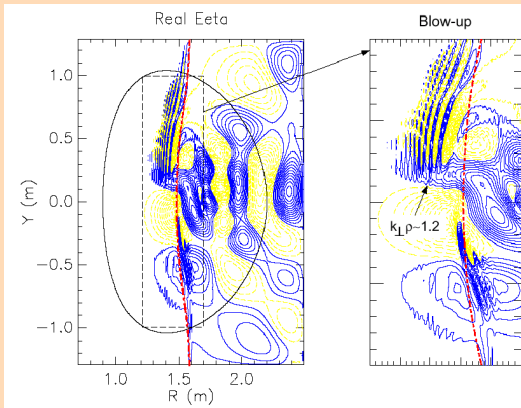


Model of a "hard" sphere included in a "soft" material, 26 million d.o.f. (Adams and Demmel, Prometheus and *PETSc*, unstructured meshes in solid mechanics).

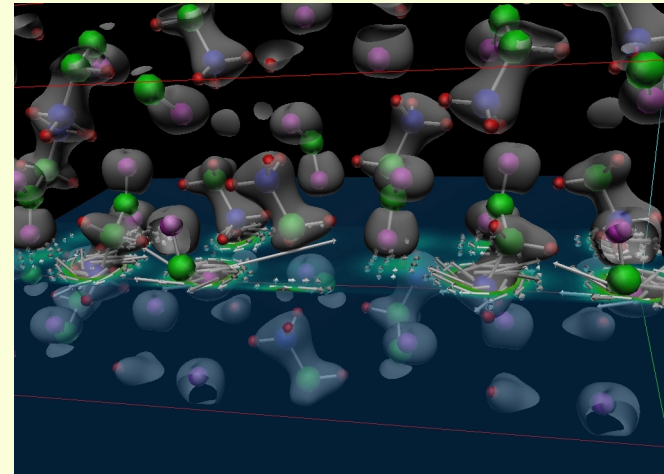


3D overlapping grid for a submarine produced with *Overture's* module *ogen*.

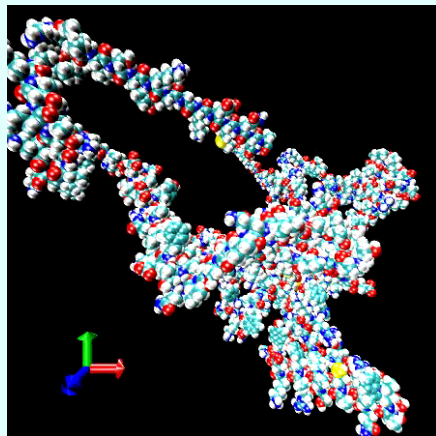
Use of ACTS Tools



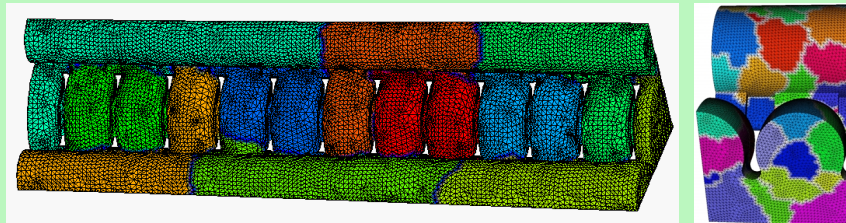
Two *ScaLAPACK* routines, *PZGETRF* and *PZGETRS*, are used for solution of linear systems in the spectral algorithms based AORSA code (Batchelor et al.), which is intended for the study of electromagnetic wave-plasma interactions. The code reaches 68% of peak performance on 1936 processors of an IBM SP.



Induced current (white arrows) and charge density (colored plane and gray surface) in crystallized glycine due to an external field (Louie, Yoon, Pfrommer and Canning), eigenvalue problems solved with *ScaLAPACK*.



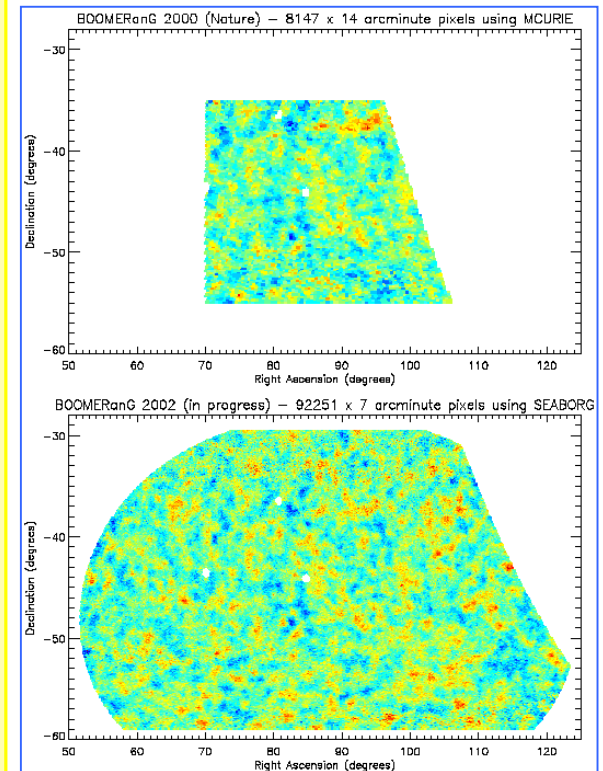
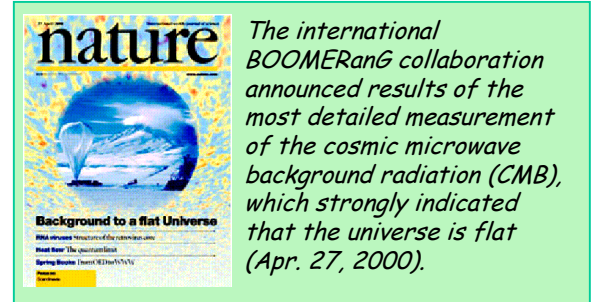
OPT++ is used in protein energy minimization problems (shown here is protein T162 from CASP5, courtesy of Meza, Oliva et al.)



Omega3P is a parallel distributed-memory code intended for the modeling and analysis of accelerator cavities, which requires the solution of generalized eigenvalue problems. A parallel exact shift-invert eigensolver based on *PARPACK* and *SuperLU* has allowed for the solution of a problem of order 7.5 million with 304 million nonzeros. Finding 10 eigenvalues requires about 2.5 hours on 24 processors of an IBM SP.

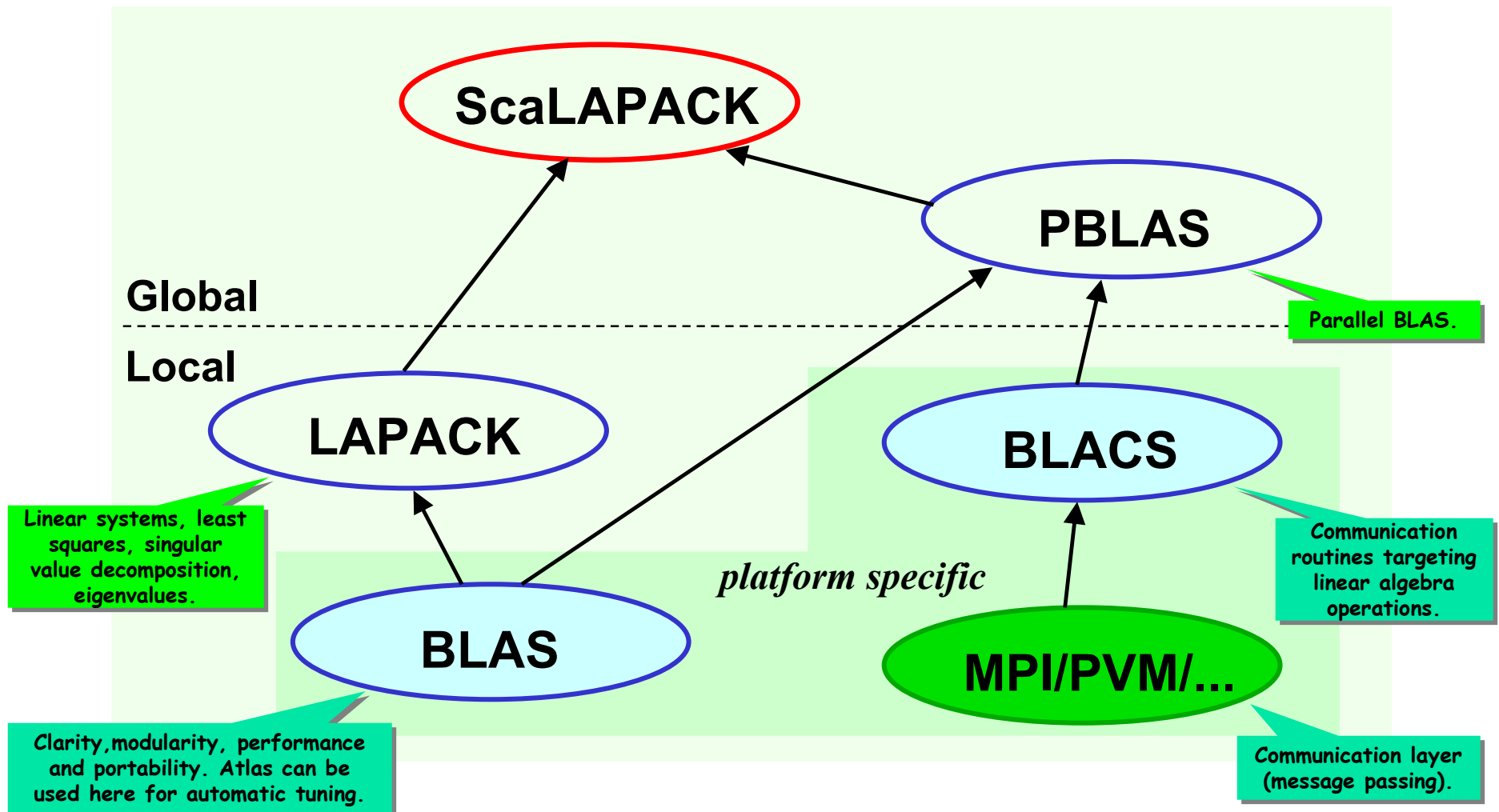
Cosmic Microwave Background (CMB) Analysis

- The statistics of the tiny variations in the CMB (the faint echo of the Big Bang) allows the determination of the fundamental parameters of cosmology to the percent level or better.
- MADCAP (Microwave Anisotropy Dataset Computational Analysis Package)
 - Makes maps from observations of the CMB and then calculates their angular power spectra. (See <http://crd.lbl.gov/~borrill>).
 - Calculations are dominated by the solution of linear systems of the form $M=A^{-1}B$ for dense $n \times n$ matrices A and B scaling as $O(n^3)$ in flops. MADCAP uses **ScaLAPACK** for those calculations.
- On the NERSC Cray T3E (original code):
 - Cholesky factorization and triangular solve.
 - Typically reached 70-80% peak performance.
 - Solution of systems with $n \sim 10^4$ using tens of processors.
 - The results demonstrated that the Universe is spatially flat, comprising 70% dark energy, 25% dark matter, and only 5% ordinary matter.
- On the NERSC IBM SP:
 - Porting was trivial but tests showed only 20-30% peak performance.
 - Code rewritten to use triangular matrix inversion and triangular matrix multiplication → one-day work
 - Performance increased to 50-60% peak.
 - Solution of previously intractable systems with $n \sim 10^5$ using hundreds of processors.



ScaLAPACK: *software structure*

<http://acts.nerisc.gov/scalapack>

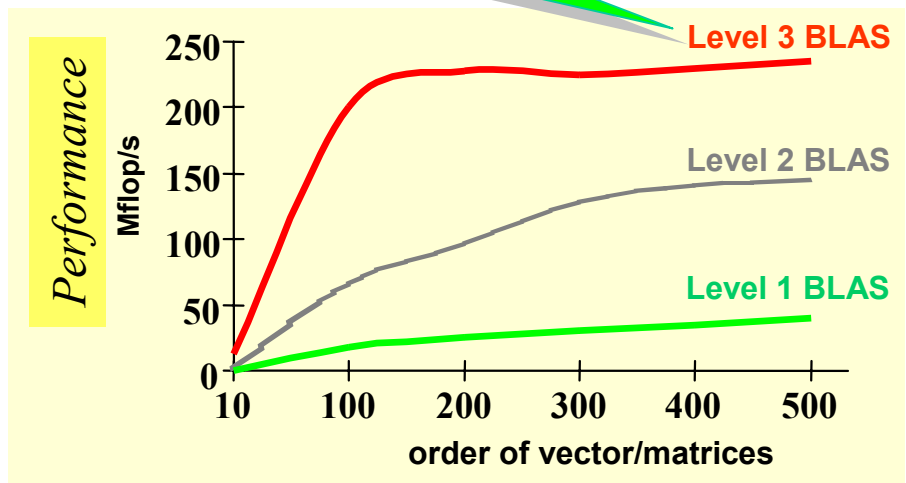


BLAS

(Basic Linear Algebra Subroutines)

- Clarity: code is shorter and easier to read.
- Modularity: gives programmer larger building blocks.
- Performance: manufacturers (usually) provide tuned machine-specific BLAS.
- Portability: machine dependencies are confined to the BLAS.
- Key to high performance: effective use of memory hierarchy (true on all architectures).

Development of blocked algorithms (BLAS 3) is important for performance!



- Level 1 BLAS: vector-vector operations.
- Level 2 BLAS: matrix-vector operations.
- Level 3 BLAS: matrix-matrix operations.

LAPACK

(<http://www.netlib.org/lapack>)

- Linear Algebra library written in Fortran 77 (Fortran 90, C and C++ versions also available).
 - Combine algorithms from LINPACK and EISPACK into a single package.
 - Efficient on a wide range of computers (RISC, Vector, SMPs).
 - User interface similar to LINPACK (Single, Double, Complex, Double Complex).
 - Built atop level 1, 2, and 3 BLAS for high performance, clarity, modularity and portability.
- Basic problems:
 - Linear systems: $Ax = b$
 - Least squares: $\min \|Ax - b\|_2$
 - Singular value decomposition: $A = U\Sigma V^T$
 - Eigenvalues and eigenvectors: $Az = \lambda z$, $Az = \lambda Bz$
 - LAPACK does not provide routines for structured problems or general sparse matrices

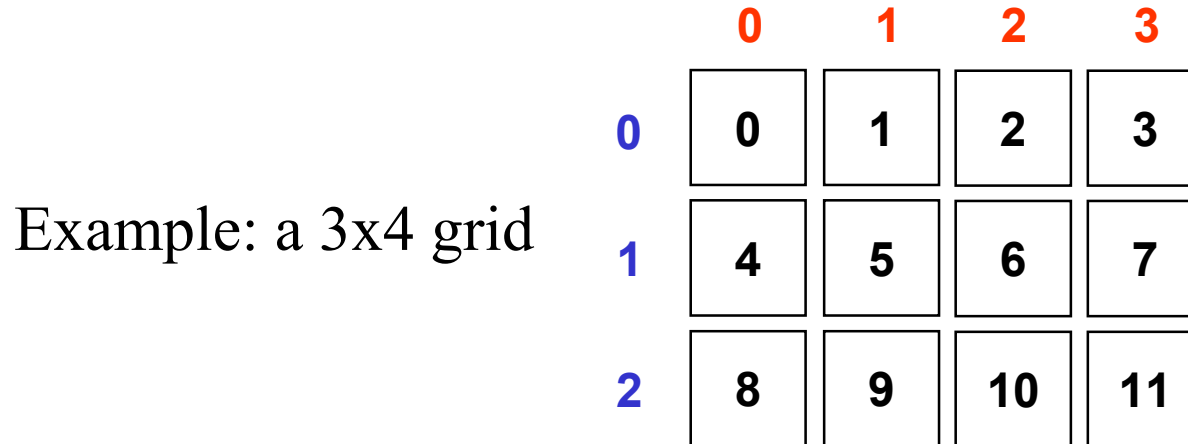
BLACS

(Basic Linear Algebra Communication Subroutines)

- A design tool, they are a conceptual aid in design and coding.
- Associate widely recognized mnemonic names with communication operations. This improves:
 - program readability
 - self-documenting quality of the code.
- Promote efficiency by identifying frequently occurring operations of linear algebra which can be optimized on various computers.

BLACS: *basics*

- Processes are embedded in a two-dimensional grid.



- An operation which involves more than one sender and one receiver is called a *scoped operation*.

Scope	Meaning
Row	All processes in a process row participate.
Column	All processes in a process column participate.
All	All processes in the process grid participate.

BLACS: *example*

```

* Get system information
  CALL BLACS_PINFO( IAM, NPROCS )
* If underlying system needs additional setup, do it now
  IF( NPROCS.LT.1 ) THEN
    IF( IAM.EQ.0 ) NPROCS = 4
    CALL BLACS_SETUP( IAM, NPROCS )
  END IF
* Get default system context
  CALL BLACS_GET( 0, 0, ICTXT )
  :
* Define 1 x (NPROCS/2+1) process grid
  NPROW = 1
  NPCOL = NPROCS / 2 + 1
  CALL BLACS_GRIDINIT( ICTXT, 'Row', NPROW, NPCOL )
  CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )
* If I'm not in the grid, go to end of program
  IF( MYROW.NE.-1 ) THEN
    IF( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
      CALL DGESD2D( ICTXT, 5, 1, X, 5, 1, 0 )
    ELSE IF( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
      CALL DGERV2D( ICTXT, 5, 1, Y, 5, 0, 0 )
    END IF
    :
    CALL BLACS_GRIDEXIT( ICTXT )
  END IF
  CALL BLACS_EXIT( 0 )
END

```

(out) uniquely identifies each process
(out) number of processes available

(in) uniquely identifies each process
(in) number of processes available

(in) integer handle indicating the context
(in) use (default) system context
(out) BLACS context

(output) process row and column coordinate

send X to process (1,0)

receive X from process (0,0)

leave context

exit from the BLACS

See <http://www.netlib.org/blacs> for more information.

- The BLACS context is the BLACS mechanism for partitioning communication space.
- A message in a context cannot be sent or received in another context.
- The context allows the user to
 - create arbitrary groups of processes
 - create multiple overlapping and/or disjoint grids
 - isolate each process grid so that grids do not interfere with each other
- BLACS context \Leftrightarrow MPI communicator

PBLAS

(Parallel Basic Linear Algebra Subroutines)

- Similar to the BLAS in portability, functionality and naming.
- Built atop the BLAS and BLACS
- Provide global view of matrix

```
CALL DGEXXX( M, N, A( IA, JA ), LDA, ... )
```

BLAS

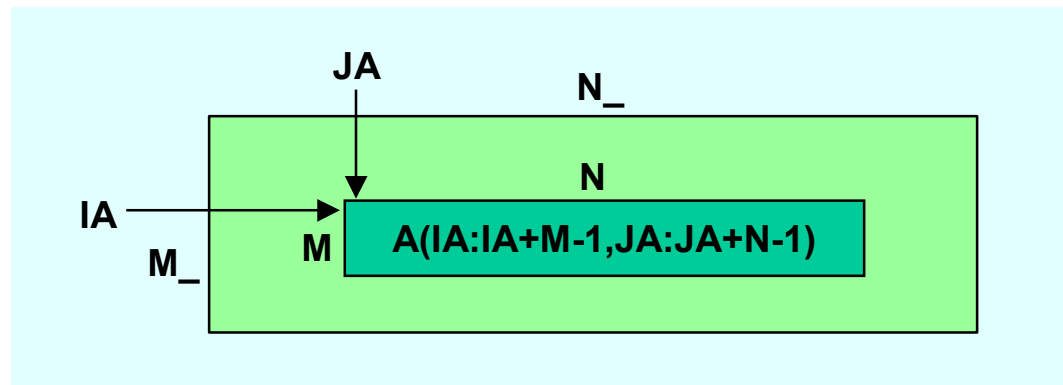
```
CALL PDGEXXX( M, N, A, IA, JA, DESCA, ... )
```

PBLAS

Array descriptor
(see next slides)

PBLAS: *levels and view of the operands*

- Levels:
 - Level 1: vector-vector operations.
 - Level 2: matrix-vector operations.
 - Level 3: matrix-matrix operations.
- Global view of the matrix operands, allowing global addressing of distributed matrices (hiding complex local indexing)

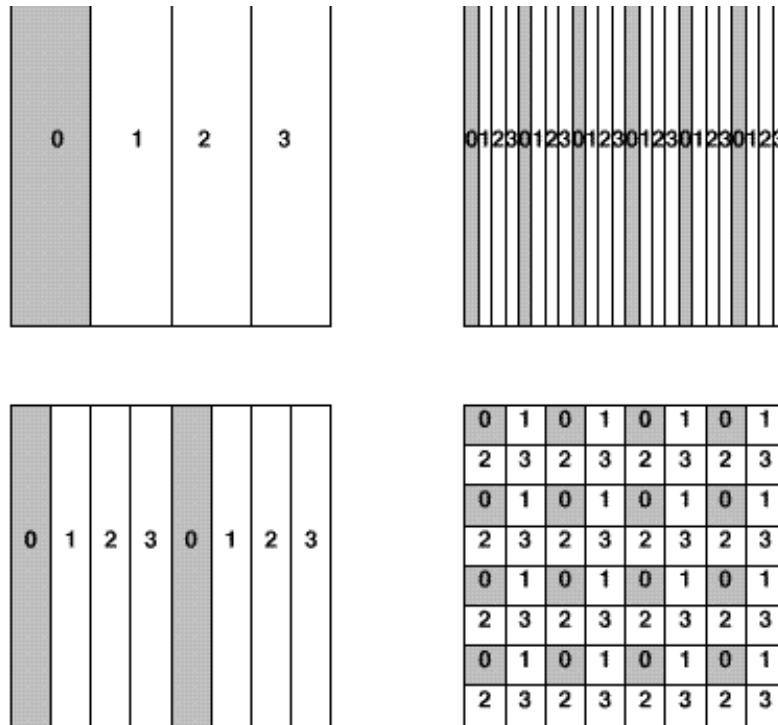


ScaLAPACK: *goals*

- Efficiency
 - Optimized computation and communication engines
 - Block-partitioned algorithms (Level 3 BLAS) for good node performance
- Reliability
 - Whenever possible, use LAPACK algorithms and error bounds
- Scalability
 - As the problem size and number of processors grow
 - Replace LAPACK algorithm that did not scale (new ones into LAPACK)
- Portability
 - Isolate machine dependencies to BLAS and the BLACS
- Flexibility
 - Modularity: build rich set of linear algebra tools (BLAS, BLACS, PBLAS)
- Ease-of-Use
 - Calling interface similar to LAPACK

ScaLAPACK: *data layouts*

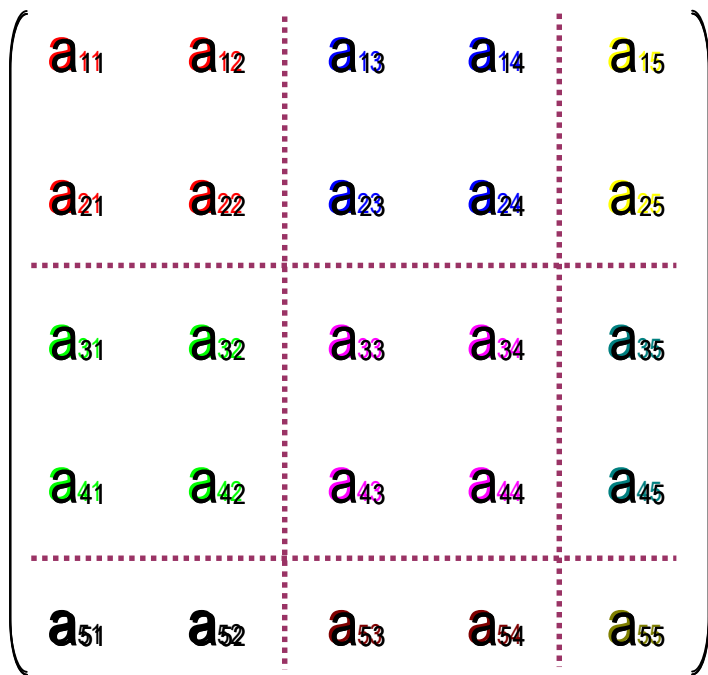
- 1D block and cyclic column distributions



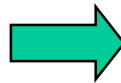
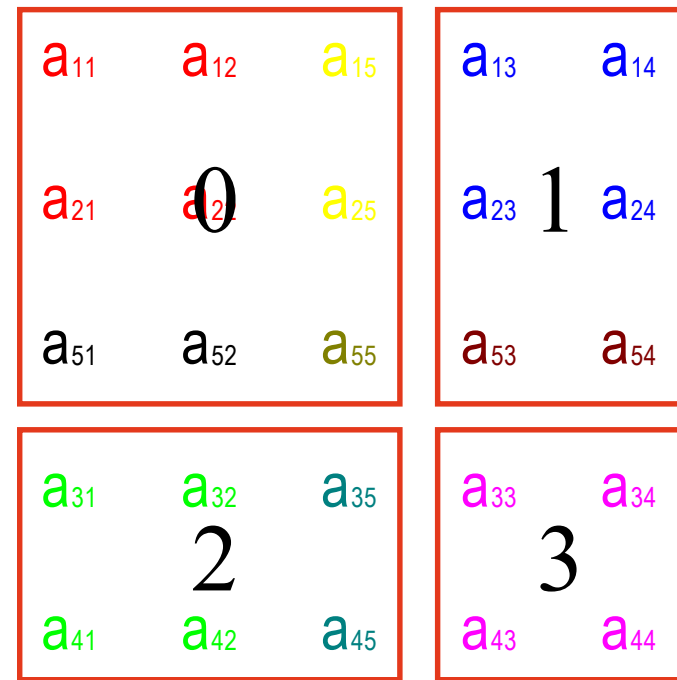
- 1D block-cycle column and 2D block-cyclic distribution
- 2D block-cyclic used in ScaLAPACK for dense matrices

ScaLAPACK: 2D Block-Cyclic Distribution

5x5 matrix partitioned in 2x2 blocks



2x2 process grid point of view



2D Block-Cyclic Distribution

1.1	1.2	1.3	1.4	1.5
-2.1	2.2	2.3	2.4	2.5
-3.1	-3.2	3.3	3.4	3.5
-4.1	-4.2	-4.3	4.4	4.5
-5.1	-5.2	-5.3	-5.4	5.5



	0	1		
	a_{11}	a_{12}	a_{15}	a_{13} a_{14}
0	a_{21}	a_{22}	a_{25}	a_{23} a_{24}
	a_{51}	a_{52}	a_{55}	a_{53} a_{54}
1	a_{31}	a_{32}	a_{35}	a_{33} a_{34}
	a_{41}	a_{42}	a_{45}	a_{43} a_{44}

```

:
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

IF ( MYROW.EQ.0 .AND. MYCOL.EQ.0 ) THEN
  A(1) = 1.1; A(2) = -2.1; A(3) = -5.1;
  A(1+LDA) = 1.2; A(2+LDA) = 2.2; A(3+LDA) = -5.2;
  A(1+2*LDA) = 1.5; A(2+3*LDA) = 2.5; A(3+4*LDA) = -5.5;
ELSE IF ( MYROW.EQ.0 .AND. MYCOL.EQ.1 ) THEN
  A(1) = 1.3; A(2) = 2.3; A(3) = -5.3;
  A(1+LDA) = 1.4; A(2+LDA) = 2.4; A(3+LDA) = -5.4;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.0 ) THEN
  A(1) = -3.1; A(2) = -4.1;
  A(1+LDA) = -3.2; A(2+LDA) = -4.2;
  A(1+2*LDA) = 3.5; A(2+3*LDA) = 4.5;
ELSE IF ( MYROW.EQ.1 .AND. MYCOL.EQ.1 ) THEN
  A(1) = 3.3; A(2) = -4.3;
  A(1+LDA) = 3.4; A(2+LDA) = 4.4;
END IF
:
CALL PDGESVD( JOBU, JOBVT, M, N, A, IA, JA, DESCA, S, U, IU,
              JU, DESCU, VT, IVT, JVT, DESCVT, WORK, LWORK,
              INFO )
:

```

LDA is the leading dimension of the local array (see next slides)

Array descriptor for A (see next slides)

ScaLAPACK: *array descriptors*

- Each global data object is assigned an *array descriptor*.
- The *array descriptor*:
 - Contains information required to establish mapping between a global array entry and its corresponding process and memory location (uses concept of BLACS context).
 - Is differentiated by the DTYPE_ (first entry) in the descriptor.
 - Provides a flexible framework to easily specify additional data distributions or matrix types.
- User must distribute all global arrays prior to the invocation of a ScaLAPACK routine, for example:
 - Each process generates its own submatrix.
 - One processor reads the matrix from a file and send pieces to other processors (may require message-passing for this).

Array Descriptor for Dense Matrices

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=1 for dense matrices.
2	CTXT_A	(global)	BLACS context handle.
3	M_A	(global)	Number of rows in global array A.
4	N_A	(global)	Number of columns in global array A.
5	MB_A	(global)	Blocking factor used to distribute the rows of array A.
6	NB_A	(global)	Blocking factor used to distribute the columns of array A.
7	RSRC_A	(global)	Process row over which the first row of the array A is distributed.
8	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
9	LLD_A	(local)	Leading dimension of the local array.

Array Descriptor for Narrow Band Matrices

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_A	(global)	Descriptor type DTYPE_A=501 for 1 x Pc process grid for band and tridiagonal matrices block-column distributed.
2	CTXT_A	(global)	BLACS context handle.
3	N_A	(global)	Number of columns in global array A.
4	NB_A	(global)	Blocking factor used to distribute the columns of array A.
5	CSRC_A	(global)	Process column over which the first column of the array A is distributed.
6	LLD_A	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored.
7	—	—	Unused, reserved.

Array Descriptor for Right Hand Sides for Narrow Band Linear Solvers

DESC_()	Symbolic Name	Scope	Definition
1	DTYPE_B	(global)	Descriptor type DTYPE_B=502 for Pr x 1 process grid for block-row distributed matrices .
2	CTXT_B	(global)	BLACS context handle.
3	M_B	(global)	Number of rows in global array B
4	MB_B	(global)	Blocking factor used to distribute the rows of array B.
5	RSRC_B	(global)	Process row over which the first row of the array B is distributed.
6	LLD_B	(local)	Leading dimension of the local array. For the tridiagonal subroutines, this entry is ignored.
7	—	—	Unused, reserved.

ScaLAPACK: *Functionality*

$Ax = b$	Simple Driver	Expert Driver	Factor	Solve	Inversion	Conditioning Estimator	Iterative Refinement
Triangular				x	x	x	x
SPD	x	x	x	x	x	x	x
SPD Banded	x		x	x			
SPD Tridiagonal	x		x	x			
General	x	x	x	x	x	x	x
General Banded	x		x	x			
General Tridiagonal	x		x	x			
Least Squares	x		x	x			
GQR			x				
GRQ			x				
$Ax = \lambda x$ or $Ax = \lambda Bx$	Simple Driver	Expert Driver	Reduction	Solution			
Symmetric	x	x	x	x			
General	x	x	x	x			
Generalized BSPD	x		x	x			
SVD			x	x			

ScaLAPACK: *Performance*

- The algorithms implemented in ScaLAPACK are scalable in the sense that the parallel efficiency is an increasing function of N^2/P (problem size per node).
- Maintaining memory use per node constant allows efficiency to be maintained (in practice, a slight degradation is acceptable).
- Use efficient machine-specific BLAS (not the Fortran 77 source code available in <http://www.netlib.gov>) and BLACS (nondebug installation).
- On a distributed-memory computer:
 - Use the right number of processors
 - Rule of thumb: $P = M \times N / 1,000,000$ for an $M \times N$ matrix, which provides a local matrix of size approximately 1000-by-1000.
 - Do not try to solve a small problem on too many processors.
 - Do not exceed the physical memory.
 - Use an efficient data distribution.
 - Block size (i.e., MB, NB) = 64.
 - Square processor grid: $P_{row} = P_{column}$.

On line tutorial: <http://acts.nersc.gov/scalapack/hands-on/main.html>

Hands-On Exercises for ScaLAPACK

ScaLAPACK Team
August 2002

Introduction

These exercises provide basic and more advanced programming instruction for writing parallel programs calling the BLACS, PBLAS, and ScaLAPACK. A basic knowledge of Fortran, parallel programming with message-passing, and MPI are assumed. Some of the exercises also require an understanding of two-dimensional block cyclic data distribution.

Detailed information on the BLACS, PBLAS, and ScaLAPACK may be found at the respective URLs:

- ♦ <http://www.netlib.org/blacs>
- ♦ <http://www.netlib.org/scalapack>
- ♦ http://www.netlib.org/scalapack/pblas_qref.html

Exercises 1 and 2 give an introduction to parallel programming with the Basic Linear Algebra Communication Subprograms (BLACS). Exercises 3, 4, and 5 provide a range of simplistic to more complex parallel programs calling ScaLAPACK and PBLAS. More example programs for ScaLAPACK can be found at <http://www.netlib.org/scalapack/examples>.

The instructions for the exercises assume that the underlying system is an IBM SP or a Cray T3E; using up to six processes that do message-passing. These example programs use MPI as the underlying message-passing layer. The version of MPI used in these examples is the version 3.0, and we assume the user has this version installed.

These hands-on exercises were prepared in collaboration with the Joint Institute for Computational Science at the University of Tennessee, based on contributions from A. YarKhan, C. Hastings, S. Blackford, C. Whaley, A. Petitet and O. Marques.

Exercise 1: [BLACS - Hello World Example](#)

Exercise 2: [BLACS - Pi Example](#)

Exercise 3: [ScaLAPACK - Example Program 1](#)

Exercise 4: [ScaLAPACK - Example Program 2](#)

Exercise 5: [PBLAS Example](#)

Addition Resources:

- ♦ [Block Cyclic Data Distribution](#)
- ♦ [Useful calling sequences](#)
- ♦ [Download all exercises](#)

[ScaLAPACK](#)

[Tools](#)

[Project](#)

[Home](#)

[Search](#)

What about tuning and performance analysis?

TAU

- Profiling of Java, C++, C, and Fortran codes
 - Detailed information (much more than *prof/gprof*)
 - Profiles for each unique template instantiation
 - Time spent exclusively and inclusively in each function
 - Start/Stop timers
 - Profiling data maintained for each thread, context, and node
 - Parallel IO Statistics for the number of calls for each profiled function
 - Profiling groups for organizing and controlling instrumentation
 - Support for using CPU hardware counters (PAPI)
 - Graphic display for parallel profiling data
 - Graphical display of profiling results (built-in viewers, interface to Vampir)
- COSY: COmpile manager Status displaY
 - FANCY: File ANd Class displaY
 - SPIFFY: Structured Programming Interface and Fancy File displaY
 - CAGEY: CAll Graph Extended displaY
 - CLASSY: CLASS hierarchY browser
 - RACY: Routine and data ACcess profile displaY
 - SPEEDY: Speedup and Parallel Execution Extrapolation DisplaY

TAU: Example 1

```

PROGRAM PSGESVDRIVER
!
!   Example Program solving Ax=b via ScaLAPACK routine PSGESV
!
!   .. Parameters ..
!
!**** a bunch of things omitted for the sake of space ****
!
!   .. Executable Statements ..
!
!   INITIALIZE THE PROCESS GRID
!
integer profiler(2)
save profiler

call TAU_PROFILE_INIT()
call TAU_PROFILE_TIMER(profiler,'PSGESVDRIVER')
call TAU_PROFILE_START(profiler)
CALL SL_INIT( ICTXT, NPROW, NPCOL )
CALL BLACS_GRIDINFO( ICTXT, NPROW, NPCOL, MYROW, MYCOL )

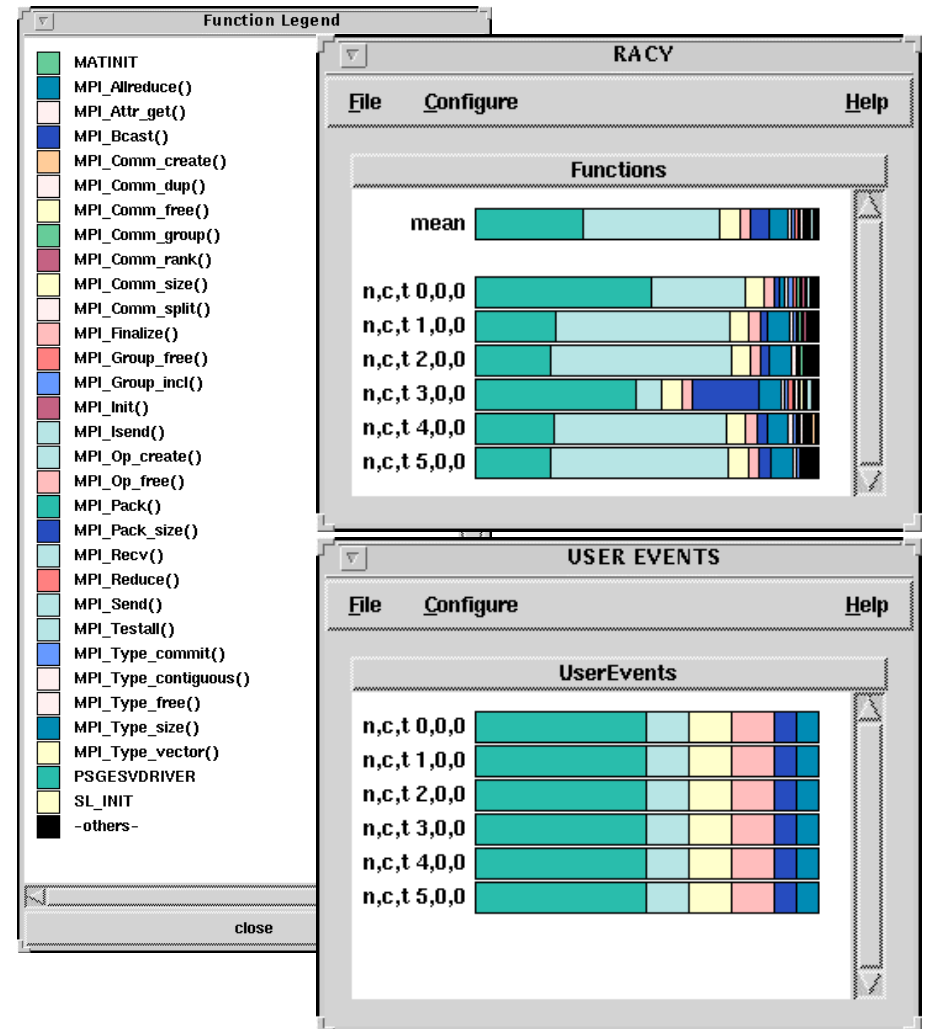
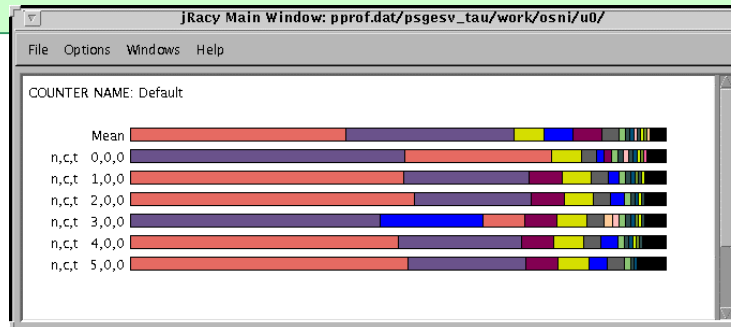
!**** a bunch of things omitted for the sake of space ****

CALL PSGESV( N, NRHS, A, IA, JA, DESCA, IPIV, B, IB, JB, DESCB, &
INFO )

!**** a bunch of things omitted for the sake of space ****

call TAU_PROFILE_STOP(profiler)
STOP
END

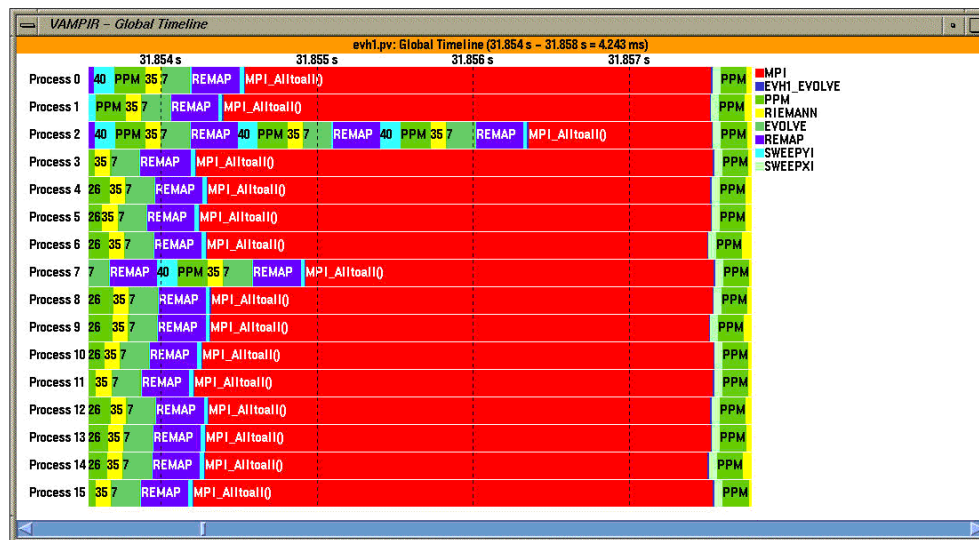
```



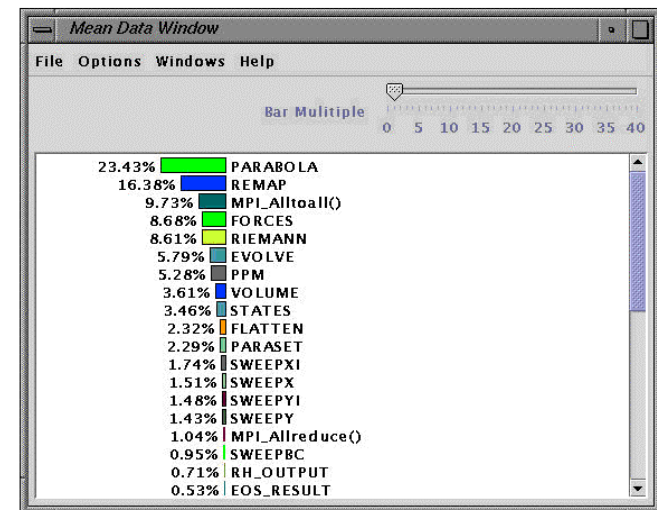
NB. ScaLAPACK routines have not been instrumented and therefore are not shown in the charts.

TAU: Example 2

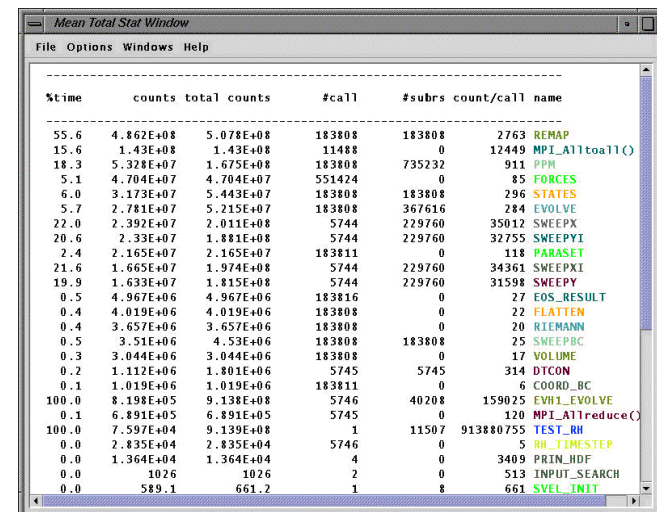
- *EVH1 (Enhanced Virginia Hydrodynamics #1) benchmark*
- *MPI code developed from VH1, based on the piece-wise parabolic method (PPM) of Colella and Woodward*
- *PPM is a technique used for compressible, non-turbulent hydrodynamics. It has been used in a variety of astrophysical contexts, in addition to some ideal gas computations and studies of convection*



Visualizing TAU traces with Vampir, a commercial trace visualization tool from Pallas, GmbH.



JRACY, time spent in each process.



JRACY, exclusive and inclusive level 1 data cache misses for all routines (except PARABOLA), mean over 16 processors.

Why is ACTS unique?

- Provides pointers and documentation about software tools.
- Accumulates the expertise and user feedback on the use of the software tools and scientific applications that used them:
 - independent software evaluations
 - participation in the developer user groups e-mail list
 - presentation of a gallery of applications
 - leverage between tool developers and tool users
 - workshops and tutorials
 - tool classification
 - support

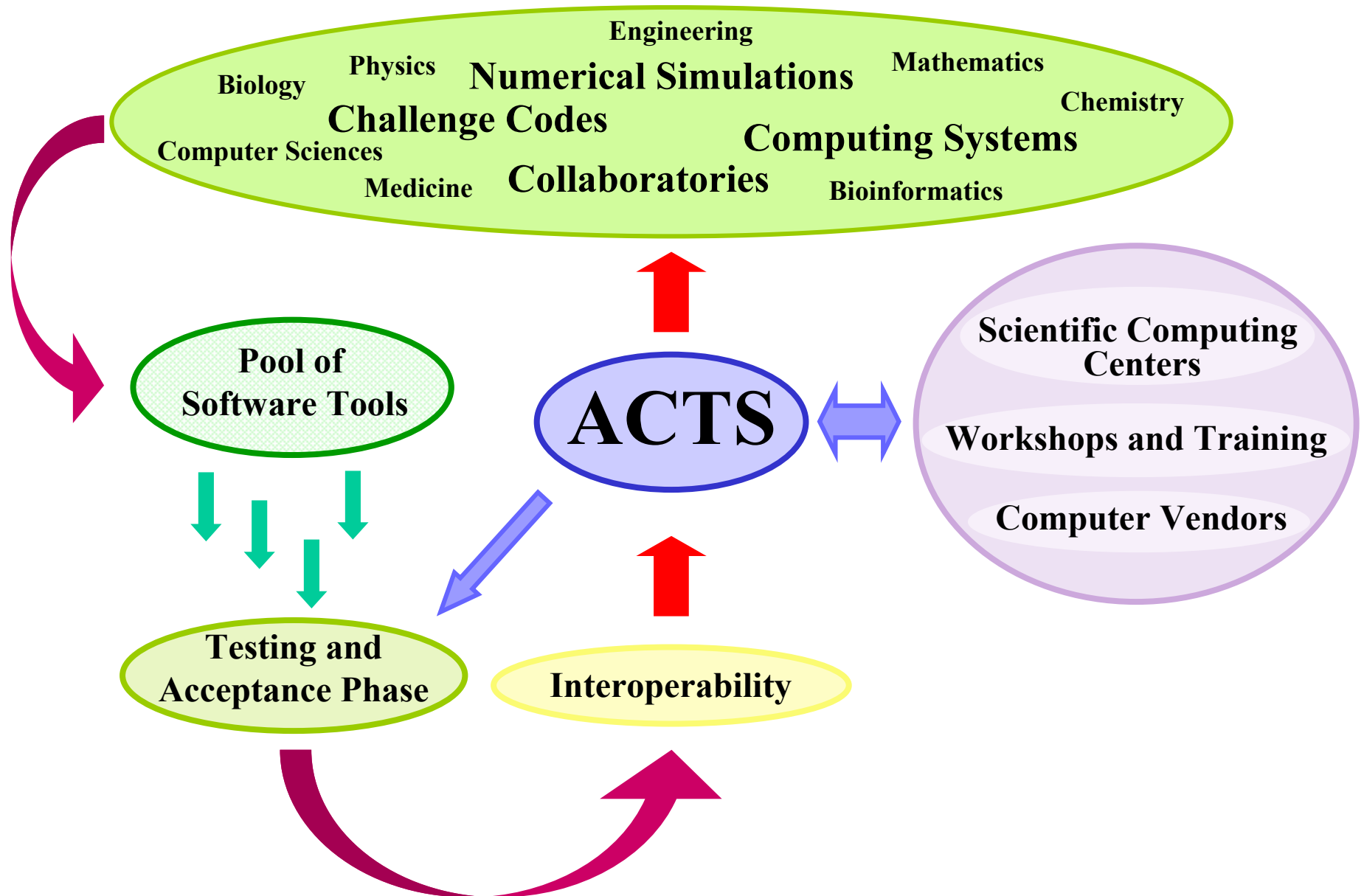
Related Activities

- Software Repositories:
 - Netlib: <http://www.netlib.org>
 - HPC-Netlib: <http://www.nhse.org/hpc-netlib>
 - National HPCC Software Exchange NHSE: <http://www.nhse.org>
 - Guide to Available Mathematical Software: <http://gams.nist.gov>
 - MGNet: <http://www.mgnet.org>
 - NEOS: <http://www-fp.mcs.anl.gov/otc/Guide>
 - OO Numerics: <http://oonumerics.org/oon>
- Portable timing routines, tools for debugging, compiler technologies:
 - Ptools: <http://www.ptools.org>
 - Center for Programming Models for Scalable Parallel Computing: <http://www.pmodels.org>
- Education:
 - Computational Science Educational Project: <http://csep1.phy.ornl.gov>
 - UCB's Applications of Parallel Computers: http://www.cs.berkeley.edu/~demmel/cs267_Spr99
 - MIT's Applied Parallel Computing: <http://www.mit.edu/~cly/18.337>
 - Dictionary of algorithms, data structures and related definitions: <http://www.nist.gov/dads>

Lessons Learned

- *There is still a gap between tool developers and application developers which leads to duplication of efforts.*
- *The tools currently included in the ACTS Collection should be seen as dynamical configurable toolkits and should be grouped into toolkits upon user/application demand.*
- *Users demand long-term support of the tools.*
- *Applications and users play an important role in making the tools mature.*
- *Tools evolve or are superseded by other tools.*
- *There is a demand for tool interoperability and more uniformity in the documentation and user interfaces.*
- *There is a need for an intelligent and dynamic catalog/repository of high performance tools.*

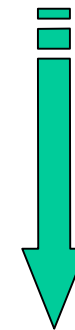
User Community



Who Benefits from these tools?

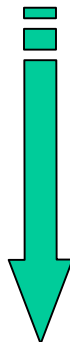
Application	Computational Problem	Software Tools	Highlights
MADCAP	Matrix factorization and triangular solves	ScaLAPACK	<ul style="list-style-type: none"> • 50% peak performance on an IBM SP • Nearly perfect scalability on 1024, 2048, 3072 and 4096 processors • Fast implementation of numerical algorithms
3-Charged Particles	Solution of large, complex unsymmetric linear systems	SuperLU	<ul style="list-style-type: none"> • Solves systems of equations of order 8.4 million on 64 processors in 1 hour of wall clock time • 30 GFLOPs
NWChem	Distribute large data arrays, collective operations	Global Arrays and LAPACK	<ul style="list-style-type: none"> • Very good scaling for large problems

<http://acts.nersc.gov/AppMat>



Enabling sciences and discoveries... with high performance and scalability...

... More Applications ...



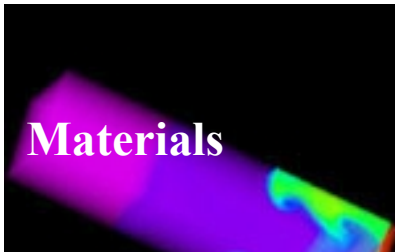
FDN3D &	Unstructured grids, compressible and incompressible Euler and Navier-Stokes equations.	PETSc	<ul style="list-style-type: none"> • Parallelization of legacy code • Gordon Bell price, 0.23 Tflop/s on 3072 procs of ASCI Red
P-FLAPW	Eigenvalue problems	ScaLAPACK	<ul style="list-style-type: none"> • Study of systems up to 700 atoms (mat size=35,000) • Runs efficiently • Facilitated the study of new problems in materials science such as impurities and disordered systems.
NIMROD	Quad and triangular high order finite elements, semi-implicit time integration, sparse matrix solvers	SuperLU	<ul style="list-style-type: none"> • Code improvement of 5 fold, equivalent to 3-5 years progress in computing hardware.



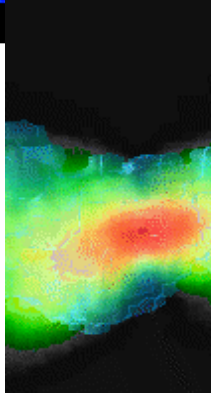
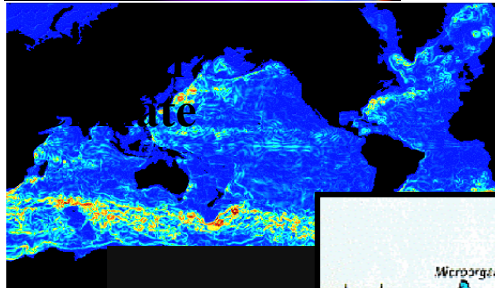
Scientific Computing – Third Pillar of Science

“a new way of doing science”

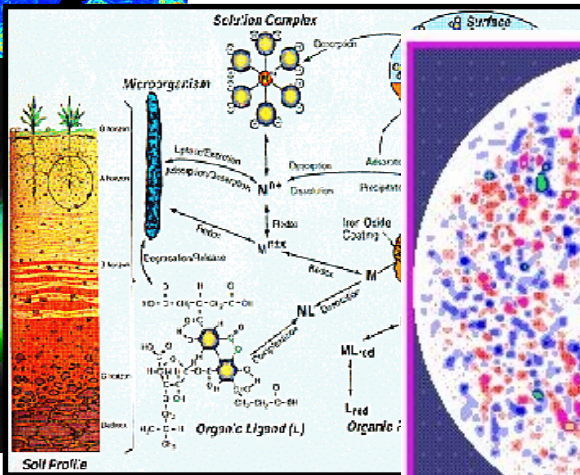
**Many SC programs
need dramatic advances
in simulation capabilities
to meet their
mission goals**



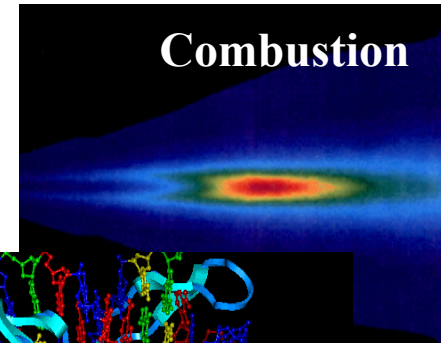
Materials



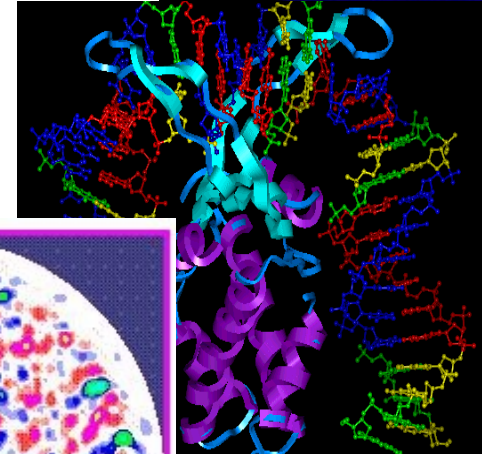
**Components
of Matter**



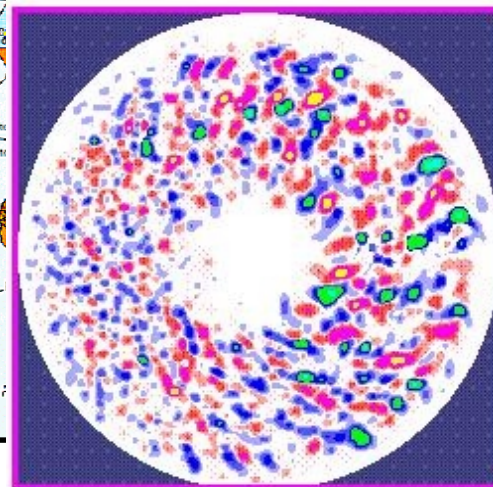
**Subsurface
Transport**



Combustion

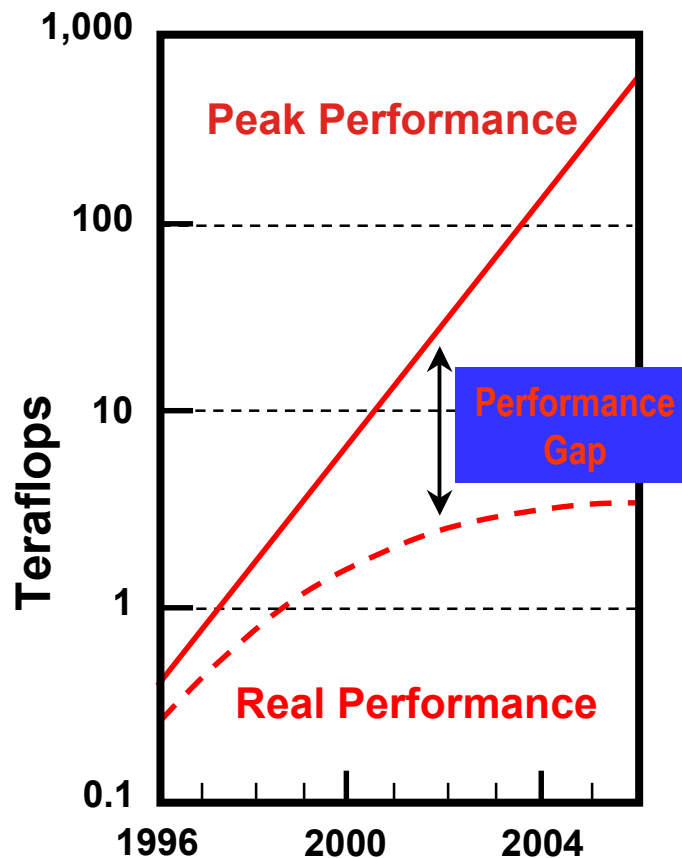
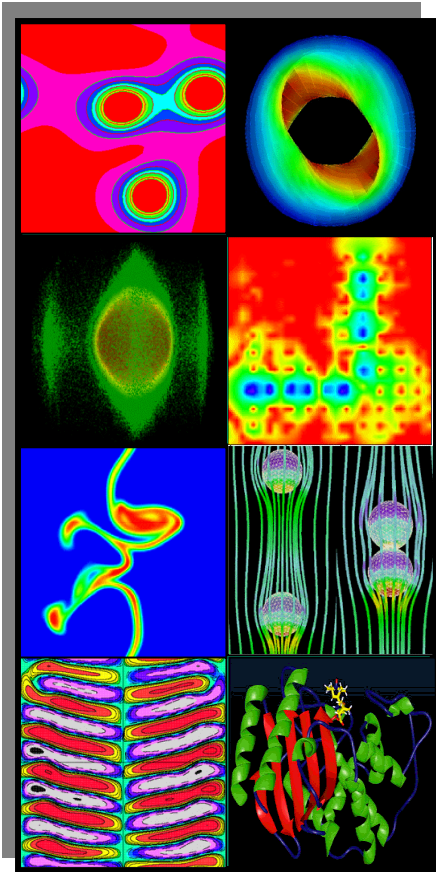


**Health Effects,
Bioremediation**



Fusion Energy

Addressing the Performance Gap through Software



Peak performance is skyrocketing

- In 1990s, peak performance increased 100x; in 2000s, it will increase 1000x

But

- Efficiency for many science applications declined from 40-50% on the vector supercomputers of 1990s to as little as 5-10% on parallel supercomputers of today

Need research on

- Mathematical methods and algorithms that achieve high performance on a single processor and scale to thousands of processors
- More efficient programming models for massively parallel supercomputers

Challenges in the Development of Scientific Codes

- Productivity
 - Time to the first solution (prototype)
 - Time to solution (production)
 - Other requirements
- Complexity
 - Increasingly sophisticated models
 - Interdisciplinarity
 - Model coupling
- Performance
 - Increasingly complex algorithms
 - Increasingly complex architectures
 - Increasingly demanding applications

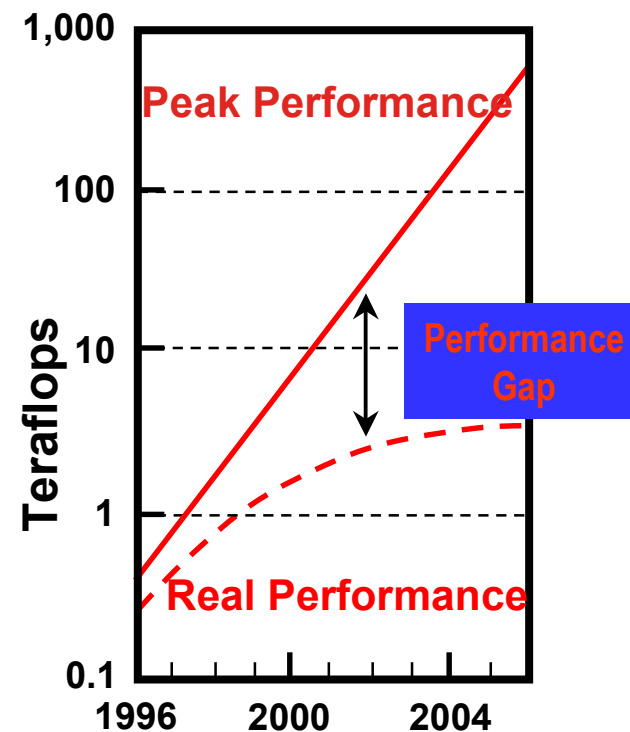
- Libraries written in different languages.
- Different pieces of the code evolve at different rates
- Swapping competing implementations of the same idea and testing without modifying the code

Peak performance is skyrocketing

- *In 1990s, peak performance increased 100x; in 2000s, it will increase 1000x*

However

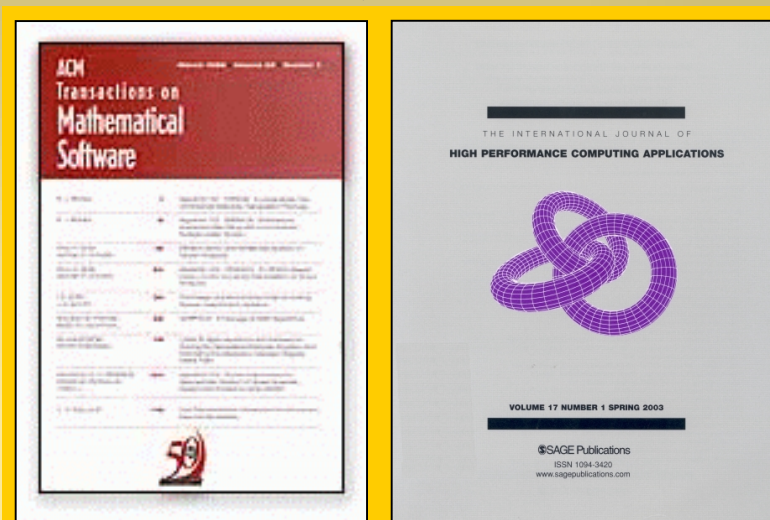
- *Efficiency for many science applications declined from 40-50% on the vector supercomputers of 1990s to as little as 5-10% on parallel supercomputers of today*



References

- An expanded Framework for the Advanced Computational Testing and Simulation Toolkit, <http://acts.nerisc.gov/documents/Proposal.pdf>
- The Advanced Computational Testing and Simulation (ACTS) Toolkit. *Technical Report LBNL-50414*.
- A First Prototype of PyACTS. *Technical Report LBNL-53849*.
- ACTS - A collection of High Performing Tools for Scientific Computing. *Technical Report LBNL-53897*.
- The ACTS Collection: Robust and high-performance tools for scientific computing. Guidelines for tool inclusion and retirement. *Technical Report LBNL/PUB-3175*.
- An Infrastructure for the creation of High End Scientific and Engineering Software Tools and Applications. *Technical Report LBNL/PUB-3176*.

To appear: two journals featuring ACTS Tools.



Progress Reports

Tutorials and Workshops

- How Can ACTS Work for you?, <http://acts.nerisc.gov/events/Workshop2000>
- Solving Problems in Science and Engineering, <http://acts.nerisc.gov/events/Workshop2001>
- Robust and High Performance Tools for Scientific Computing, <http://acts.nerisc.gov/events/Workshop2002>
- Robust and High Performance Tools for Scientific Computing, <http://acts.nerisc.gov/events/Workshop2003>
- The ACTS Collection: Robust and High Performance Libraries for Computational Sciences, SIAM PP04 <http://www.siam.org/meetings/pp04>
- New Methods for Developing Peta-scalable Codes, PSC http://www.psc.edu/training/PPS_May04


- FY 2002, <http://acts.nerisc.gov/documents/Report2002.pdf>
- FY 2003-1, <http://acts.nerisc.gov/documents/Report2003-1.pdf>
- FY 2003-2, <http://acts.nerisc.gov/documents/Report2003-2.pdf>




acts-support@nerisc.gov
<http://acts.nerisc.gov>

<http://acts.nersc.gov>

See also: <http://acts.nersc.gov/documents>



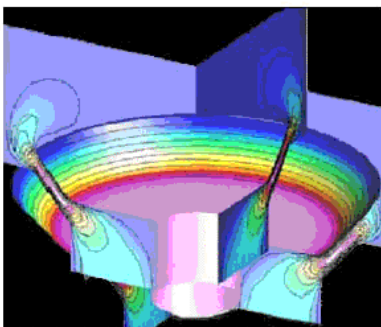
The DOE ACTS Collection



[Tools](#) [News](#) [Project](#) [Center](#) [Search](#)

The DOE ACTS (Advanced Computational Software) Collection is a set of DOE-developed software tools that make it easier for programmers to write high performance scientific applications for parallel computers. This site is the central information center for the ACTS Collection and is brought to you by NERSC and the [Mathematical, Information, and Computational Sciences](#) (MICS) Division of DOE. Correspondence regarding the collection (including requests for support) should be directed to acts-support@nersc.gov.

click on the image below to see other applications that have benefited from ACTS Tools



The image shows pressure and velocity around a moving valve in a diesel engine. The flow here was found as part of a CFD effort to simulate the flow within the complex 3D geometry of a diesel engine. The computation was carried out using the Overture Framework and the PADRE library for parallel data distribution.

[Tools](#) [News](#) [Project](#) [Center](#) [Search](#)

Tool descriptions,
installation
details, examples,
etc

Agenda,
accomplishments,
conferences,
releases, etc

Goals and other
relevant information

Points of
contact

Search
engine

- High Performance Tools
 - portable
 - library calls
 - robust algorithms
 - help code optimization
- Scientific Computing Centers like NERSC:
 - Reduce user's code development time that sums up in more production runs and faster and effective scientific research results
 - Overall better system utilization
 - Facilitate the accumulation and distribution of high performance computing expertise
 - Provide better scientific parameters for procurement and characterization of specific user needs